ABSTRACT

Title of dissertation: DEEP LEARNING FOR FASHION AND FORENSICS

Xintong Han
Doctor of Philosophy, 2018

Dissertation directed by: Professor Larry S. Davis
Department of Electrical
and Computer Engineering

Deep learning is the new electricity, which has dramatically reshaped people's everyday life. In this thesis, we focus on two emerging applications of deep learning - fashion and forensics.

The ubiquity of online fashion shopping demands effective search and recommendation services for customers. To this end, we first propose an automatic spatially-aware concept discovery approach using weakly labeled image-text data from shopping websites. We first fine-tune GoogleNet by jointly modeling clothing images and their corresponding descriptions in a visual-semantic embedding space. Then, for each attribute (word), we generate its spatially-aware representation by combining its semantic word vector representation with its spatial representation derived from the convolutional maps of the fine-tuned network. The resulting spatially-aware representations are further used to cluster attributes into multiple groups to form spatially-aware concepts (*e.g.*, the *neckline* concept might consist of attributes like *v-neck*, *round-neck*, *etc*). Finally, we decompose the visual-semantic

embedding space into multiple concept-specific subspaces, which facilitates structured browsing and attribute-feedback product retrieval by exploiting multimodal linguistic regularities. We conducted extensive experiments on our newly collected Fashion200K dataset, and results on clustering quality evaluation and attribute-feedback product retrieval task demonstrate the effectiveness of our automatically discovered spatially-aware concepts.

For fashion recommendation tasks, we study two types of fashion recommendation: (i) suggesting an item that matches existing components in a set to form a stylish outfit (a collection of fashion items), and (ii) generating an outfit with multimodal (images/text) specifications from a user. To this end, we propose to jointly learn a visual-semantic embedding and the compatibility relationships among fashion items in an end-to-end fashion. More specifically, we consider a fashion outfit to be a sequence (usually from top to bottom and then accessories) and each item in the outfit as a time step. Given the fashion items in an outfit, we train a bidirectional LSTM (Bi-LSTM) model to sequentially predict the next item conditioned on previous ones to learn their compatibility relationships. Further, we learn a visual-semantic space by regressing image features to their semantic representations aiming to inject attribute and category information as a regularization for training the LSTM. The trained network can not only perform the aforementioned recommendations effectively but also predict the compatibility of a given outfit. We conduct extensive experiments on our newly collected Polyvore dataset, and the results provide strong qualitative and quantitative evidence that our framework outperforms alternative methods.

In addition to searching and recommendation, customers also would like to virtually try-on fashion items. We present an image-based VIirtual Try-On Network (VITON) without using 3D information in any form, which seamlessly transfers a desired clothing item onto the corresponding region of a person using a coarse-to-fine strategy. Conditioned upon a new clothing-agnostic yet descriptive person representation, our framework first generates a coarse synthesized image with the target clothing item overlaid on that same person in the same pose. We further enhance the initial blurry clothing area with a refinement network. The network is trained to learn how much detail to utilize from the target clothing item, and where to apply to the person in order to synthesize a photo-realistic image in which the target item deforms naturally with clear visual patterns. Experiments on our newly collected dataset demonstrate its promise in the image-based virtual try-on task over state-of-the-art generative models.

Interestingly, VITON can be modified to swap faces instead of swapping clothing items. Conditioned on the landmarks of a face, generative adversarial networks can synthesize a target identity on to the original face keeping the original facial expression. We achieve this by introducing an identity preserving loss together with a perceptually-aware discriminator. The identity preserving loss tries to keep the synthesized face presents the same identity as the target, while the perceptually-aware discriminator ensures the generated face looks realistic. It is worth noticing that these face-swap techniques can be easily used to manipulated people's faces, and might cause serious social and political consequences.

Researchers have developed powerful tools to detect these manipulations. In

this dissertation, we utilize convolutional neural networks to boost the detection accuracy of tampered face or person in images. Firstly, a two-stream network is proposed to determine if a face has been tampered with. We train a GoogLeNet to detect tampering artifacts in a face classification stream, and train a patch based triplet network to leverage features capturing local noise residuals and camera characteristics as a second stream. In addition, we use two different online face swapping applications to create a new dataset that consists of 2010 tampered images, each of which contains a tampered face. We evaluate the proposed two-stream network on our newly collected dataset. Experimental results demonstrate the effectiveness of our method.

Further, spliced people are also very common in image manipulation. We describe a tampering detection system containing multiple modules, which model different aspects of tampering traces. The system first detects faces in an image. Then, for each detected face, it enlarges the bounding box to include a portrait image of that person. Three models are fused to detect if this person (portrait) is tampered or not: (i) PortraintNet: A binary classifier fine-tuned on ImageNet pre-trained GoogLeNet. (ii) SegNet: A U-Net predicts tampered masks and boundaries, followed by a LeNet to classify if the predicted masks and boundaries indicating the image has been tampered with or not. (iii) EdgeNet: A U-Net predicts the edge mask of each portrait, and the extracted portrait edges are fed into a GoogLeNet for tampering classification. Experiments show that these three models are complementary and can be fused to effectively detect a spliced portrait image.

DEEP LEARNING FOR FASHION AND FORENSICS

by

Xintong Han

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Larry S. Davis Chair/Advisor
Professor Rama Chellappa
Professor Joseph F. JaJa
Professor Min Wu
Professor David W. Jacobs

© Copyright by
Xintong Han
2018

Dedication

This thesis is dedicated to my parents, Mingwu Han and Huiying Huang, as well as my grilfriend Mengxi Wang, for their love and support.

Acknowledgments

First, I would like to thank my country, China, for shaping my culture and my personality.

My greatest gratitudes goes to all the people who have made this thesis possible and because of whom my Ph.D. experience has been one that I will cherish in the rest of my life.

I'd like to thank my advisor, Professor Larry Davis, for giving me an invaluable opportunity and freedom to work on challenging and interesting computer vision projects over the past five years. He has always made himself available for help and advice. It has been such a pleasure to work with and learn from such an extraordinary advisor.

I would also like to thank my colleagues at the computer vision lab, who have enriched my graduate life in many ways and deserve a special mention. Bharat Singh helped me understand various deep learning theories and good practices, I especially enjoyed the time we exchanged our ideas on interesting research projects. Zhe Wu brought me into the world of Hearthstone, when I felt tired after a long time of coding, playing some Hearthstone games with him was always the most relaxing and joyful experience. Zuxuan Wu helped me write several of my papers, his deep understanding of paper writing and presentation has been one of the most important reasons that our co-authored papers were accepted. Peng Zhou, as another smart colleague who worked closely with me, co-authored several interesting papers and inspired me a lot with his brilliant mind. I also enjoyed and will cherish the time

Zuxuan, Peng and I spent at the gym. Additionally, my interaction with Vlad Morariu, Hao Zhou, Yaming Wang, Ang Li, Ruichi Yu has been very fruitful.

I acknowledge the help and support from some of my roommates. I would like to express my gratitude to Yu Jin, Guanzhong Wang, Li Bai, Jun Wang for their friendship and the food they shared with me.

I owe my deepest thanks to my family - my mother and father who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them. I thank my girlfriend, Mengxi Wang, for her patience and understanding.

Lastly, I would like to say thank you to all the people who help me survive when pursing my Ph.D.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction and Motivation

Recent years have witnessed the increasing demands of online shopping for fashion items. Online apparel and accessories sales in US are expected to reach 123 billion in 2022 from 72 billion in 2016 [4]. Despite the convenience online fashion shopping provides, a major cost in online clothing shopping is the high return rate of nearly 50%. This can be effectively addressed by better recommendation and browsing systems, and virtual try-on techniques. In this dissertation, we will first describe the deep learning tools we developed to tackle these problems.

In Chapter 2, we introduce a spatially-aware concept discovery approach to automatically group attributes describing the same fashion characteristics into clusters. The discovered concepts can be used to project a large collection of fashion items onto a 2D plane based on customized criteria for structured product browsing. Moreover, this approach also enables attribute-feedback product retrieval that can help customers to refine their initial search results.

Chapter 3 studies the problem of fashion recommendation. In contrast to existing methods that mainly utilize a Siamese network to model the compatibility of two items [5, 6], we consider an outfit as a sequence of fashion items and use an RNN-type model to model multiple items' compatibility beyond just focusing

on pairs. The proposed method can not only recommend suitable items based on existing ones provided by the users, but can also adaptively adjust the results conditioned on multimodal specifications.

The last fashion application we will present is an image-base virtual try-on system without using an 3D information (Chapter 4). We introduce a coarse-to-fine framework. In the coarse stage, we first leverage a generative model to produce photo-realistic new image by overlaying a product image onto the corresponding region of a clothed person. This synthesized image is then refined by warping the target clothing and compose the warped image to the coarse result. Our approach generates realistic and appealing virtual try-on results and outperforms state-of-the-art image synthesis methods.

At the same time, deep learning also leads to sophisticated tools for image and video manipulation [7–9]. In Chapter 5, we present how to use a Generative Adversarial Network to change the identity of faces (face-swap). To preserve the desired identity, we inject target identity in an encoder-decoder generator. And we further propose a more powerful discriminator to obtain more realistic face-swap results.

However, techniques like this will cause serious social and political consequences if not use in appropriate scenarios. In Chapter 6 of this dissertation, we will introduce several models that leverage both low-level and high-level information for determining if a face or person in an image is a result of a manipulation.

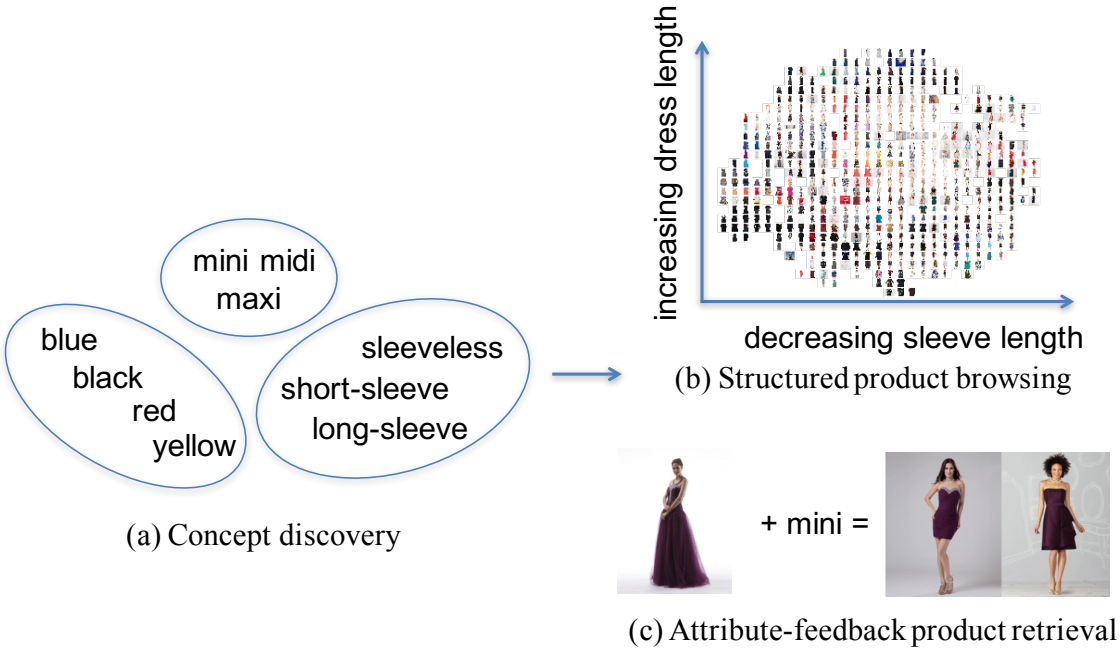
Finally, we summarize this dissertation and discuss on potential research directions in Chapter 7.

Chapter 2: Automatic Spatially-aware Fashion Concept Discovery

2.1 Introduction

The exponential growth of online fashion shopping websites has encouraged techniques that can effectively search for a desired product from a massive collection of clothing items. However, this remains a particularly challenging problem since, unlike generic objects, clothes are usually subject to severe deformations and demonstrate significant variations in style and texture, and, most importantly, the long-standing semantic gap between low-level visual features and high-level intents of customers is very large. To overcome the difficulty, researchers have proposed interactive search to refine retrieved results with humans in the loop. Given candidate results, customers can provide various feedback, including the relevance of displayed images [10, 11], or tuning parameters like color and texture, and then results are updated correspondingly. However, relevance feedback is limited due to its slow convergence to meet the customer requirements. In addition to color and texture, customers often wish to exploit higher-level features, such as *neckline*, *sleeve length*, *dress length*, *etc.*

Semantic attributes [12], which have been applied effectively to object categorization [13, 14] and fine-grained recognition [15] could potentially address such



(a) Concept discovery

(b) Structured product browsing

(c) Attribute-feedback product retrieval

Figure 2.1: (a) We propose a concept discovery approach to automatically cluster spatially-aware attributes into meaningful concepts. The discovered spatially-aware concepts are further utilized for (b) structured product browsing (visualizing images according to selected concepts) and (c) attribute-feedback product retrieval (refining search results by providing a desired attribute).

challenges. They are mid-level representations that describe semantic properties. Recently, researchers have annotated clothes with semantic attributes [16–20] (*e.g.*, material, pattern) as intermediate representations or supervisory signals to bridge the semantic gap. However, annotating semantic attributes is costly. Further, attributes conditioned on object parts have achieved good performance in fine-grained recognition [21,22], confirming that spatial information is critical for attributes. This also holds for clothing images. For example, the *neckline* attribute usually corre-

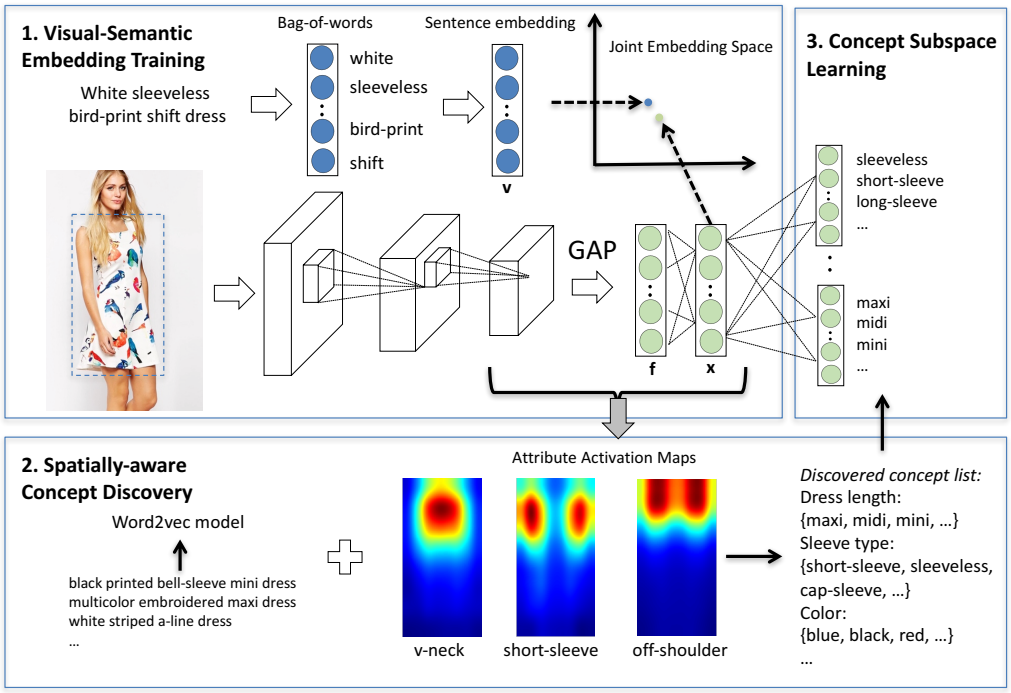


Figure 2.2: Overview of our approach. Our approach mainly contains three parts: 1. Joint embedding space training. A joint visual-semantic embedding space is trained using clothing images and their product descriptions. 2. Spatially-aware concept discovery. We use neural activations provided by global pooling (GAP) layer to generate attribute activation maps (AAMs) of attributes. The AAM captures the spatial information of attributes (*i.e.*, what is the spatial location an attribute usually refers to). By combining attributes' spatial information and their semantic representations obtained from a word2vec model, we cluster attributes into concepts. 3. Concept subspace learning. For each discovered concept, we further train a sub-network to effectively measure the similarity of images according to this concept only.

sponds to the top part in images while the *sleeve* attribute ordinarily relates to the left and right side of images.

To address the above limitations, we jointly model clothing images and their product descriptions with a visual-semantic embedding, and propose a novel approach that automatically discovers spatially-aware concepts, each of which is a collection of attributes describing the same characteristic (*e.g.*, if the concept is *color* then the attributes could contain *yellow* and *blue*, as shown in Figure 2.1(a)). In addition, we learn a subspace embedding for each discovered concept to facilitate a structured exploration of the dataset based on the concept of interest (Figure 2.1(b)). More importantly, inspired by [23], we leverage the learned visual-semantic space to exploit multimodal linguistic regularities for attribute-feedback product retrieval. For example, an image of a "white sleeveless dress" − "sleeveless" + "long-sleeve" would be near images of "white long-sleeve dress". In contrast to [23] which requires explicitly specifying the attribute to remove, we implicitly remove corresponding attributes based on the discovered concepts (Figure 2.1(c)).

Figure 2.2 provides an overview of the framework. Specifically, our framework contains the following three steps (1) we first train a joint visual-semantic embedding space using clothing images and their product descriptions. Given an image, we compute its features with GoogleNet, which are further projected into the embedding space to minimize the distance to its product description encoded by bag-of-words of attributes. By fine-tuning GoogleNet in an end-to-end fashion, we train a discriminative model that contains localization information of attributes; (2) we then obtain the spatial representation for each attribute, indicating where

in images the attribute mostly corresponds to, from the attribute activation maps. These spatial representations are further utilized to augment their corresponding semantic word representations (word vectors) produced from a skip-gram model. Further, clustering is performed to discover concepts, each of which contains semantically related attributes (*e.g.*, *maxi*, *midi*, *mini* are all different *dress length*); (3) we further disentangle the trained visual-semantic embedding by training a subspace embedding for each discovered concept, in which the similarities among items can be measured based on the corresponding concept only. The transformation of images into a subspace embedding facilitates attribute-feedback clothing search and structured browsing of fashion images.

Given the fact that existing datasets only contain images and annotated attributes (which are often very sparse) rather than image and product description pairs, we constructed the Fashion200K dataset, which contains more than 200,000 clothing images of five categories (dress, top, pants, skirt and jacket) and their associated product descriptions from online shopping websites. These five classes are the most important verticals in fashion due to their various styles and occasions. Thus, we focus on these categories in our dataset, but our method is applicable to any fashion categories. We conduct extensive experiments on this dataset to validate the efficacy of the automatically discovered concepts in attribute-feedback product retrieval as well as structured fashion image browsing.

Our main contributions are two-fold. First, we demonstrate that the augmentation of semantic word vectors for attributes with their spatial representations can be used to effectively cluster attributes into semantically meaningful and spatially-

aware concepts. Second, we leverage semantic regularities in the visual-semantic space for attribute-feedback clothing retrieval.

2.2 Related Work

Interactive image search. Extensive studies have been conducted on interactive image search, aiming to improve retrieved results from search engines with user feedback [10, 11, 20] (See [24] for a comprehensive review). The basic idea is to refine the results by incorporating feedback from users, including the relevance of the candidates, and tuning low-level parameters like color and texture. In practice, relevance feedback requires a large number of iterations to converge to user intent. Also, it requires manual annotations to define the relative attributes, which limits its scalability. In addition, when searching clothing images, customers generally focus on certain higher-level characteristics, such as *neckline*, *sleeve length*, *etc.*, thus rendering relevance feedback less useful.

Attributes for clothing modeling. There have been numerous works focusing on utilizing semantic attributes as mid-level representations for clothing modeling. For instance, Chen *et al.* [17] learned semantic attributes for clothing on the human upper body. Huang *et al.* [18] built tree-structured layers for all attribute categories to form a semantic representation for clothing images. Veit *et al.* [25] learned visually relevant semantic subspaces using a multi-query triplet network. Kovashka *et al.* [20] utilized relative attributes with ranking functions instead of using binary feedback for retrieval tasks. In contrast, we propose a novel concept

discovery framework, in which a concept is a collection of automatically identified attributes derived by jointly modeling image and text.

Visual concept discovery. To exploit the substantial amounts of weakly labeled data, researchers have proposed various approaches to discover concepts. Sun *et al.* [26] combined visual and semantic similarities of concepts to cluster concepts while ensuring their discrimination and compactness. Vittayakorn *et al.* [27] and Berg *et al.* [28] verified the visualness of attributes, and [27] also uses neural activations to learn the characteristics of each attribute. Vaccaro *et al.* [29] utilized a topic model to learn latent concepts and retrieve fashion items based on textual specifications. Singh *et al.* [30] discovered pair-concepts for event detection and discard irrelevant concepts by the co-occurrences of concepts. Recently, some works discovered the spatial extents of concepts. Xiao and Lee [31] discovered visual chains for locating the image regions that are relevant to one attribute. Singh and Lee [32] introduced a deep network to jointly localize and rank relative attributes. However, these approaches involve training a single model for each individual attribute, which is not scalable.

Visual-semantic joint embedding. Our work is also related to visual-semantic embedding models [23, 33–36]. Frome *et al.* [33] recognize objects with a deep visual-semantic embedding model. Kiros *et al.* [23] adopted an encoder-decoder framework coupled with a contrastive loss to train a joint visual-semantic embedding. Wang *et al.* [34] combined cross-view ranking loss and within-view structure preservation loss to map images and their descriptions. Beyond training a joint visual-semantic embedding with image and text pairs as in these works, we further

decompose the trained embedding space into multiple concept-specific subspaces, which facilitates structured browsing and attribute-feedback product retrieval by exploiting multimodal linguistic regularities.

2.3 Fashion200K Dataset

There have been several clothing datasets collected recently [18, 19, 36–38]. However, none of these datasets are suitable for our task because they do not contain descriptions of images. This prevents us from learning semantic representations for attributes using word2vec [39]. Thus, we collected the Fashion200K dataset and automatically discover concepts from it.

We first crawled more than 300,000 product images and their product descriptions from online shopping websites and removed the ones whose product descriptions contain fewer than four words, resulting in over 200,000 images. We then split them into 172,049 images for training, 12,164 for validation, and 25,331 for testing. For cleaning product descriptions, we deleted stop words, symbols, as well as words that occur fewer than 5 times. Each remaining word is regarded as an attribute. Finally, there are 4,404 attributes for training the joint embedding.

Example clothing image and description pairs are shown in Figure 2.3. Since we wish to automatically discover concepts from this noisy dataset and learn concept-level subspace features, we do not conduct any manual annotations for this dataset. Note that as a preprocessing step, we trained a detector using the MultiBox model [40] for all five categories and run them on all images. Only the detected foregrounds

Figure 2.3: Examples of the image-text pairs in Fashion200K.

are cropped and used as input to our model.

2.4 Fashion Concept Discovery

In this section, we present the key components of the proposed concept discovery approach shown in Fig. 2.2, including visual-semantic embedding learning, spatially-aware concept discovery and concept subspace learning. Since our method leverages spatial information of an attribute, and the same attribute in different types of clothing (*e.g.*, "short" in "short dress" and "short pants") will have different spatial characteristics, we train an individual model for each category in our dataset. For simplicity in notation and illustration, we only show the concept discovery approach for dresses, while the same pipeline is applied to other categories in the same fashion. Results of all categories are shown and evaluated in our experiments.

2.4.1 Visual-semantic Embedding

To fully explore the substantial weakly labeled web data for mining concepts, we first train a joint visual-semantic embedding model with image-text pairs by projecting a product image and its associated text into a joint embedding space. Following [23], we also utilize a stochastic bidirectional contrastive loss to achieve good convergence.

More formally, let $\mathbf{I}$ denote an image and $S = \{w_1, w_2, ..., w_N\}$ its corresponding text, where w_i is the i-th attribute (word) in the product description. Let $\mathbf{W_I}$ denote the image embedding matrix, and $\mathbf{W_T}$ denote the attribute embedding matrix. We first represent the i-th word w_i with one-hot vector $\mathbf{e}_i$, which is further encoded into the embedding space by $\mathbf{v}_i = \mathbf{W_T} \cdot \mathbf{e}_i$. We then represent the product description with bag-of-words $\mathbf{v} = \frac{1}{N}\sum_i \mathbf{v_i}$. Similarly, for the image $\mathbf{I}$, we first compute its feature vector $\mathbf{f} \in \mathbb{R}^{2048}$ with a GoogleNet model [41] parameterized by weights $\mathbf{V}$ after the global average pooling (GAP) layer as shown in Figure 2.2. Then we project the feature vector into the embedding space, in which the original image is represented as $\mathbf{x} = \mathbf{W_I} \cdot \mathbf{f}$.

The similarity between an image and its description is computed with *cosine* similarity, *i.e.*, $d(\mathbf{x}, \mathbf{v}) = \mathbf{x} \cdot \mathbf{v}$, where $\mathbf{x}$ and $\mathbf{v}$ are normalized to unit norm. Finally, the joint embedding space is trained by minimizing the following contrastive loss:

$$\begin{aligned} \min_{\mathbf{\Theta}} \sum_{\mathbf{x},k} \max(0, m - d(\mathbf{x}, \mathbf{v}) + d(\mathbf{x}, \mathbf{v}_k)) + \\ \sum_{\mathbf{v},k} \max(0, m - d(\mathbf{v}, \mathbf{x}) + d(\mathbf{v}, \mathbf{x}_k)), \end{aligned} \tag{2.1}$$

where $\mathbf{\Theta} = \{\mathbf{W_I}, \mathbf{W_T}, \mathbf{V}\}$ contains the parameters to be optimized, and $\mathbf{v}_k$ denotes non-matching descriptions for image $\mathbf{x}$ while $\mathbf{x_k}$ are non-matching images for description $\mathbf{v}$. By minimizing this loss function, the distance between $\mathbf{x}$ and its corresponding text $\mathbf{v}$ is forced to be smaller than the distance from unmatched descriptions $\mathbf{v_k}$ by some margin m. Vice versa for description $\mathbf{v}$. In this dissertation, the feature vector $\mathbf{f}$ is extracted with the GoogleNet model [41] after the global average pooling (GAP) layer as shown in Figure 2.2.

2.4.2 Spatially-aware Concept Discovery

The training process of a joint visual-semantic embedding will lead to a discriminative CNN model, which contains not only the semantic information (*i.e.*, the last embedding layer) but also important spatial information that is hidden in the network. We now discuss how to obtain spatially-aware concepts from the network.

Attribute spatial representation. Spatial information of an attribute is crucial for understanding what part of a clothing item the attribute refers to. Motivated by [42], we generate embedded attribute activation maps (EAAM), which can localize the salient regions of attributes for an image by a single forward pass with the trained network.

Given an image $\mathbf{I}$, let $q_k(i, j)$ be the activation of unit k in the last convolutional layer at location (i, j). After the global average pooling (GAP) operation, $f_k = \sum_{i,j} q_k(i, j)$ is the k-th dimension feature of the image representation $\mathbf{f}$. For a given attribute a, the cosine distance $d(\mathbf{x}, \mathbf{W}^a)$ between image embedding $\mathbf{x}$ and attribute

embedding $\mathbf{W}^a$ indicates the probability that attribute a is present in this image. If we plug f_k into the cosine distance we obtain:

$$\begin{aligned} d(\mathbf{x}, \mathbf{W}^a) &= \sum_m W_m^a x_m = \sum_m W_m^a \sum_k W_{I_{m,k}} f_k \\ &= \sum_m W_m^a \sum_k W_{I_{m,k}} \sum_{i,j} q_k(i,j) \\ &= \sum_{i,j} \sum_m W_m^a \sum_k W_{I_{m,k}} q_k(i,j) \end{aligned} \tag{2.2}$$

where W_m^a and $W_{I_{m,k}}$ are entries of the attribute embedding $\mathbf{W}^a$ and image embedding matrix $\mathbf{W_I}$, respectively. Thus, the embedded attribute activation map (EAAM) for attribute a of image $\mathbf{I}$ can be defined as:

$$\mathbf{M}_a^{\mathbf{I}}(i,j) = \sum_m W_m^a \sum_k W_{I_{m,k}} q_k(i,j) \tag{2.3}$$

Since $d(\mathbf{x}, \mathbf{W}^a) = \sum_{i,j} \mathbf{M}_a^{\mathbf{I}}(i,j)$, $\mathbf{M}_a^{\mathbf{I}}(i,j)$ indicates how likely the attribute appears at spatial location (i,j).

Figure 2.4 shows sample EAAMs of images. We can see the activation maps indicate where the joint embedding model looks to identify an attribute. Product images on shopping websites usually have clean backgrounds and are displayed in an aligned frontal view. Thus, for a particular attribute a and its positive training set (*i.e.*, images whose product descriptions contain a) P_a, we average EAAMs for all images in P_a to generate an activation map A_a. We refer to it as the attribute activation map (AAM) of a:

$$\mathbf{A}_a = \frac{1}{|P_a|} \sum_{\mathbf{I} \in P_a} \mathbf{M}_a^{\mathbf{I}} \tag{2.4}$$

Figure 2.4: Embedding attribute activation map for a given attribute. The generated activation maps successfully highlight the discriminative regions for the given attribute.

Figure 2.5 shows AAMs of some attributes for the dress category. From this figure, we can discover that for attributes that have clear spatial information in a dress image, their AAMs capture the spatial patterns. For example, *belt* is most likely to occur in the middle part of dress images, *long-sleeve* often occurs on two sides of dress images, and *off-shoulder* is around the shoulder region of a dress. However, for some attributes whose locations are not certain for different dress images, like *floral*, *stripe*, and colors, their AAMs span almost the entire image.

Therefore, for each attribute in a clothing category, its AAM can serve as

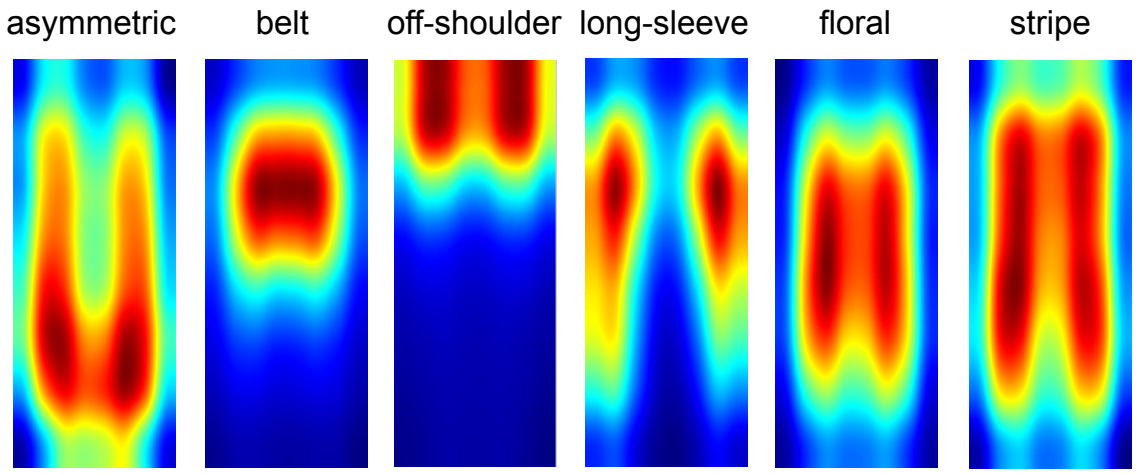


Figure 2.5: Attribute activation map for a given attribute of the dress category. The most frequency locations an attribute corresponds to in an image are highlighted.

a spatial representation. If two attributes describe the similar spatial part of a clothing category, *e.g.*, *sleeveless* and *long-sleeve*, or *v-neck* and *mockneck*, their spatial information should also be similar.

Attribute semantic representation. Only using spatial information is not sufficient for effective concept discovery, especially for those attributes that do not have a discriminative spatial representation. Thus, we train a skip-gram model [39] on the descriptions of clothing products to obtain the semantic representations (Word2vec vectors) for all attributes in our dataset. We denote the semantic representation of attribute a as $\mathbf{E}_a$.

Attribute clustering. Ideally, attributes belonging to the same concept describe the same characteristic of a clothing category; that means they should be both spatially consistent and semantically similar. Thus, for an attribute a, by simply flattening its spatial representation $\mathbf{A}_a$ and concatenating it with its semantic

concepts discovered by our method	
dress	*dress length*: maxi, midi, mini
	neckline: v, plunge, deep, high, scoop
	shoulder: off-the-shoulder, one-shoulder, strapless, ...
top	*decoration*: lace, embellished, embroidered, beaded, ...
	sleeve length: sleeveless, long-sleeve, short-sleeve, ...
	sleeve shape: kimono, cap, dolman, bell, flutter, ...
pants	*color*: black, blue, multicolor, gray, white, green, ...
	pant cut: straight-leg, slim-leg, tapered-leg, bootcut, ...
	pattern: check, geometric, leopard, palm, abstract, ...

Table 2.1: Concept discovered by our method. Each row contains the attributes belong to one concept. Ellipsis is used when the attribute list is too long to show.

representation $\mathbf{E}_a$, we can generate a feature vector:

$$\mathbf{F}_a = [vec(\mathbf{A}_a), \mathbf{E}_a] \tag{2.5}$$

where $vec(\cdot)$ is vectorization operation and we normalize $vec(\mathbf{A}_a)$ and $\mathbf{E}_a$ to have unit norm before concatenation. As a result, this attribute feature is aware of the spatial information of the attribute and can also capture its semantic meaning. K-means clustering algorithm is then used to cluster all the attributes into attribute groups, such that the attributes within a group form a concept. Unlike [26], we do not directly use visual similarity between attributes because attributes describing the same characteristic might be visually dissimilar. For example, *blue* and *red* are

both color attributes, but they are visually very different.

Table 2.1 presents some concepts discovered by our method for different categories. We find that the attributes describing the same characteristic are grouped into one cluster. For example, all attributes describing colors are in one concept because they are very close in the semantic embedding space (they are often the first word in product descriptions) and their AAMs do not provide much useful information (the right two AAMs in Figure 2.5). Thus, the semantic representations of those attributes dominate in this case and place them in the same concept. Different kinds of sleeves also form a concept, since their AAMs are very similar (along with the two sides of dresses or tops) and their word vectors are also close. We also observe that our method can successfully group noisy (not visually perceptible) attributes together, because the semantic and spatial information of these attributes is not discriminative. These noisy clusters will be discovered by our method and not affect the attribute-feedback, since customers will not provide an attribute with low visualness for retrieval. We will further evaluate the quality of the discovered concepts in the experiments.

2.4.3 Concept Subspace Learning

The discovered concepts are further utilized to refine the learned joint visual-semantic space, so that similarities between items can be measured by each individual concept (*e.g.*, *color* and *neckline* could result in different similarities). This is crucial for cases when customers want to modify attributes to refine the search

results or hope to browse products based on a particular concept. Therefore, given the concepts discovered by the attribute clustering process, we further train a sub-network for each concept, constructing a concept-specific subspace.

For a concept $C = \{a_1, a_2, ..., a_n\}$ where a_i is an attribute in this concept, we build a fully-connected layer and a softmax layer on top of the image embedding features to classify the a_i. The number of neurons in the softmax layer is $n+1$ (each attribute corresponds to one neuron with an additional one for none-of-above). This network is trained only on images with a_i in their product descriptions plus a small number of randomly sampled negative images. We denote $\mathbf{S}^C(\mathbf{x})$ to be the softmax output of the sub-network for concept C given the input image $\mathbf{x}$.

After the subspace training stage, the concept subspace features (hidden layer representations) are aware of the attributes of this particular concept, and hence enable the similarity measurement among images based only on this concept. For example, a "blue maxi dress" is more similar to a "blue mini dress" than a "red maxi dress" in the *color* subspace. However, a "red maxi dress" is closer to "blue maxi dress" in *dress length* subspace. As a result, customers can choose the desired similarity measure during online shopping so they can better explore the clothing gallery.

2.4.4 Attribute-feedback Product Retrieval

Based on the discovered concepts and learned concept subspaces, we leverage multimodal linguistic regularities to help perform attribute-feedback product

retrieval task. Some example results can be found in Figure 2.7.

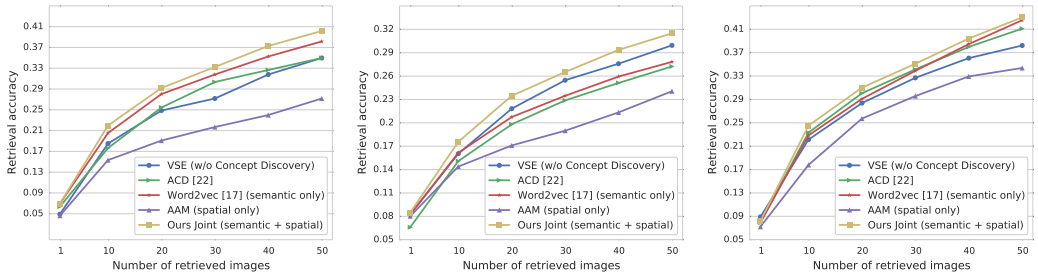
Given a retrieved image ("red sleeveless mini dress", for example), customers may want to change one attribute of the image while keeping others fixed, say "I want this dress to have long-sleeves". As we already trained a visual-semantic embedding (VSE), a baseline method would be sorting database images based on their cosine distances with the query image + query attribute (*long-sleeve*). In this way, the retrieved images have a high score for the query attribute and are similar to the query image at the same time. For a query image $\mathbf{x_q}$ and a query attribute $\mathbf{w_p}$, the attribute-feedback retrieval task to find image $\mathbf{x_o}$ is defined as:

$$\mathbf{x_o} = \arg\max_{\mathbf{x}} \ (\mathbf{x_q} + \mathbf{w_p}) \cdot \ \mathbf{x} \tag{2.6}$$

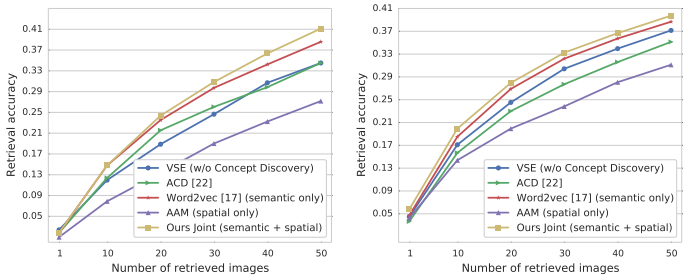
However, one problem with this approach is that it retrieves images which are closest to "red sleeveless long-sleeve mini dress" instead of "red long-sleeve mini dress". To overcome this, we note that by providing a query attribute, customers implicitly intend to remove an existing attribute (*sleeveless* in this case) that describes the same characteristic of the product as the query attribute. Since the attributes within one discovered concept describe the same characteristic, we detect the implicit negative attribute $\mathbf{w_n}$ and use it to search image $\mathbf{x_o}$:

$$\begin{aligned} \mathbf{w_n} &= \arg\max_{\mathbf{w} \in C} \ \mathbf{S}^C(\mathbf{x_q}) \\ \mathbf{x_o} &= \arg\max_{\mathbf{x}} \ (\mathbf{x_q} + \mathbf{w_p} - \mathbf{w_n}) \cdot \ \mathbf{x} \end{aligned} \tag{2.7}$$

where C is the concept to which $\mathbf{w_p}$ belongs and $\mathbf{S}^C(\mathbf{x_q})$ is the softmax output of the sub-network for C. Thus, $\mathbf{w_n}$ is the attribute in C that is most likely to be



(a) dress (b) top (c) pants



(d) skirt (e) jacket

Figure 2.6: Top-k retrieval accuracy of different methods for attribute-feedback product retrieval for dresses, tops, pants, skirt, and jacket.

present in the query image $\mathbf{x_q}$. By subtracting the detected negative attribute $\mathbf{w_n}$ from the query embedding, we remove the negative attribute to avoid two visually contradictory attributes (*e.g.*, *sleeveless* and *long-sleeve*) hurting the retrieval performance. Eqn. 2.7 indicates that our method actually uses multimodal linguistic regularities [23] with automatic negative attribute detection.

Because the subspace networks are trained with a *none-of-above* class, it might predict that $\mathbf{x_q}$ does not have any attributes in concept C. In this case, our method degenerates to the baseline method.

2.5 Experimental Results and Discussions

2.5.1 Experiment Setup

Clothing detection. Some works have shown that using detected clothing segmentations instead of entire images can achieve better performance in various tasks [18, 38], so we also train a detector for each clothing category using MultiBox model [40] to detect and crop clothing items in our dataset. Because the product images on shopping websites have clean backgrounds, the detectors work very well.

Visual-semantic embedding. We use GoogleNet InceptionV3 model [41] for the image CNN. Its global average pooling (GAP) layer after the last convolutional layer enables us to directly use it without changing the structure of the network as in [42]. We use the 2048D features right after GAP as the image features. The dimension of the joint embedding space is set to 512, thus $\mathbf{W_I}$ is a 2048×512 matrix, and $\mathbf{W_T}$ is an $M \times 512$ matrix, where M is the number of attributes. We set the margin $m = 0.2$ in Eqn. 2.1. The initial learning rate is 0.05 and is decayed by a factor of 2 after every 8 epochs. The batch size is set to 32. Finally, we fine-tune all layers of the network pretrained on ImageNet.

Spatiallly-aware concept discovery. The feature map size of the last convolutional layer in the InceptionV3 model is $8 \times 8 \times 2048$, hence the attribute activation map is of size 8×8. After vectorizing the activation map, an attribute will have a 64D feature vector as its spatial representation. We also set the dimension of word vectors to 64 to have the same dimentionality when training the Word2vec [39]

model. The window size is set to 1 because a product name is usually around 5-8 words, we do not want the attributes in one product name to use the attributes in other product names as positive context to train the word vectors. During concept clustering, instead of using all attributes, we only use 300 attributes that occur most frequently for each category, because we need enough positive training images for AAM to have a meaningful spatial representation. Also, the lower-ranked attributes often have lower visualness (e.g., *reversible*, *more*, *london*) or have the same meaning as higher-ranked attributes (e.g., *longsleeved* vs *long-sleeve*, *stripe* vs *stripes*). In the experimental results, we find that even only using the top 300 attributes can achieve better retrieval performance than the model without the proposed concept discovery approach. The number of clusters is fixed to 50.

Subspace feature learning. We set the hidden layer of each concept subspace to have 128 neurons. The learning rate is fixed to be 0.1 and we stop training after 10 epochs. Note that during training subspace networks, the visual-semantic embedding weights are fixed, only the parameters after the image embedding layer are updated.

2.5.2 Evaluation of Discovered Concepts

To evaluate the quality of our discovered concepts, a fashion professional manually assigned around 300 attributes into different categories (*e.g.*, *color*, *pattern*, *neckline*, *sleeve*, *etc.*). We use this information as ground truth concept assignments of the attributes and compare our approach with the following methods: Automatic

	Homogeneity	Completeness	V-measure
ACD [26]	0.770	0.527	0.626
Word2vec	0.765	0.534	0.629
Ours AAM	0.680	0.447	0.540
Ours Joint	**0.794**	**0.561**	**0.658**

Table 2.2: Comparison among concept discovery methods. Homogeneity, completeness and V-measure [1] are between 0 and 1, higher is better.

Concept Discovery (ACD) [26], only using semantic representations of attributes for clustering (Word2vec [39]) and only using spatial information (Our AAM). In all methods, we set the number of clusters to 50. Homogeneity, completeness and V-measure [1] are used to evaluate the clustering quality.

Results are shown in Table 2.2. Only using semantic information gives reasonable results. However, just relying on spatial information performs worst, since for many attributes, their spatial information is not discriminative and thus fails to discover informative concepts. ACD performs similarly to Word2vec because it combines semantic and visual similarities of attributes but visually dissimilar attributes may also describe the same characteristic. By jointly clustering the semantic and spatial representations of attributes, our concept discovery approach outperforms other methods by 0.03 in V-measure.

2.5.3 Attribute-feedback Product Retrieval

To evaluate how the discovered concepts help attribute-feedback product retrieval, we collected 3,167 product pairs from the test set. The two products in each pair have one attribute that differs in their product descriptions, *e.g.*, "blue geometric long-sleeve shirt" vs. "blue paisley long-sleeve shirt", "blue off-shoulder floral dress" vs. "blue one-shoulder floral dress", *etc.* In each pair, we use the image of one product and the differing attribute in their descriptions as the query to retrieve the images of the other product. Top-k retrieval accuracy is used for evaluation.

As shown in Figure 2.6, we compare our full method for all five categories with other methods. We also include the baseline method (VSE w/o concept discovery as in Eqn. 2.6), where no negative attribute is used.

We can see that using only attribute activation maps (AAM) significantly reduces performance of retrieval due to lack of semantic information. Only using semantic information (Word2vec) helps for most categories, but is worse than the baseline when retrieving tops. By adding visual information, ACD performs slightly worse than Word2vec because the visual similarity of attributes is not suitable for discovering concepts. After combining both semantic and spatial information, our concept discovery approach achieves the highest retrieval accuracy for all five categories, especially for the categories top, dress and jacket whose attributes have strong spatial information (*e.g.*, *collar shape*, *sleeve length*, *sleeve shape*). However, for clothing items like pants, whose attributes do not present informative spatial cues, our method only yields a marginal improvement over Word2vec.

Figure 2.7: Examples of our attribute-feedback product retrieval results. *Sleeve type* changes from *sleeveless* to *cap-sleeve* in the first example, and *shoulder* changes from *one-shoulder* to *strapless* in the third example, according to customer feedback attributes. The attributes in parentheses are the negative attributes automatically detected by our method.

Figure 2.7 illustrates some examples which show that our retrieval model can accurately detect the negative attribute and give satisfying results with the desired attributes added to the original results.

2.5.4 Structured Browsing of Products

Figure 2.8, 2.9 use t-SNE [43] to visualize two subspace embeddings based on two discovered concepts. In Figure 2.8, the subspace network is trained to distinguish {maxi, midi, mini} for dresses, and it learns a continuous representation of the

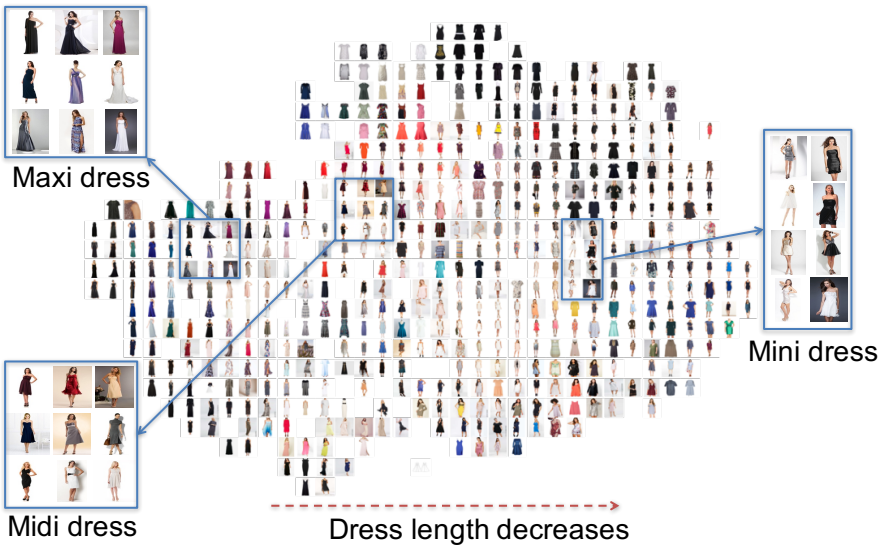


Figure 2.8: Subspace embedding corresponding to concept {maxi, midi, mini} for dresses. Images are mapped to a grid for better visualization.

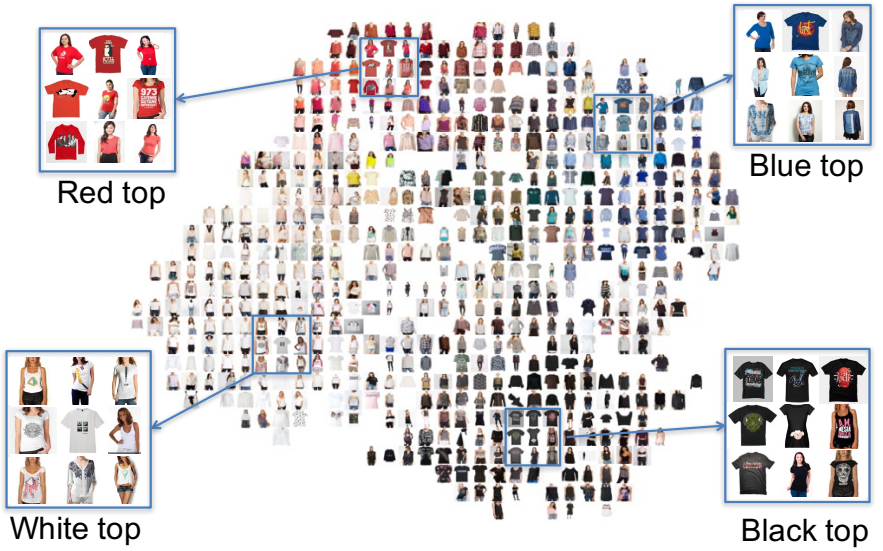


Figure 2.9: Subspace embedding corresponding to concept {black, blue, white, red, gray, green, purple, beige, ...} for tops.

length of dresses - dress length decreases from left to right on the 2D visualization plane. Figure 2.9 illustrates the embedding corresponding to the attributes describing colors for tops. Tops with different colors are well separated in the embedding subspace. Although Veit *et al.* [25] also learns concept subspaces based on an attention mechanism, they heavily rely on richly annotated data, while our method is fully automatic and annotation free.

By training a subspace embedding for each discovered concept, we can project images into the appropriate subspace and explore the images according to this specific concept, while a general embedding (like the visual-semantic embedding) cannot automatically adjust its representations based on user-specified characteristics.

Thus, the subspace features enable structured browsing during online shopping. For example, when a customer finds a mini dress and wants to see other dresses that share similar length with this dress, she may choose the subspace of {maxi, midi, mini}, so she can find the other mini dresses near her initial choice and as she explores the left side of the subspace, she can find dresses with longer length.

We should note that it is also possible to concatenate subspace embeddings of two concepts, hence clothing items sharing the same characteristics according to two concepts will be close in the concatenated subspace.

2.6 Conclusion

We automatically discover spatially-aware concepts with clothing images and their product descriptions. By projecting images and their attributes into a joint

visual-semantic embedding space, we are able to learn attribute spatial representations. We then combine spatial representations and semantic representations of attributes, and cluster attributes into spatially-aware concepts, such that the attributes in one concept describe the same characteristic. Finally, a subspace embedding is trained for each concept to capture the concept-specific information. Experiments on clustering quality evaluation and attribute-feedback product retrieval for five clothing categories show the effectiveness of the discovered concepts and the learned subspace features.

Chapter 3: Learning Fashion Compatibility with Bidirectional LSTMs

3.1 Introduction

Fashion plays an increasingly significant role in our society due to its capacity for displaying personality and shaping culture. Recently, the rising demands of online shopping for fashion products motivate techniques that can recommend fashion items effectively in two forms (1) suggesting an item that fits well with an existing set and (2) generating an outfit (a collection of fashion items) given text/image inputs from users. However, these remain challenging problems as they require modeling and inferring the compatibility relationships among different fashion categories that go beyond simply computing visual similarities.

Extensive studies have been conducted on automatic fashion analysis in the multimedia community. However, most of them focus on clothing parsing [44, 45], clothing recognition [19], or clothing retrieval [46, 47]. Although, there are a few works that investigated fashion recommendation [46, 48, 49], they either fail to consider the composition of items to form an outfit [46] or only support one of the two recommendation categories discussed above [48, 49]. In addition, it is desirable that recommendations can take multimodal inputs from users. For example, a user can provide keywords like "business", or an image of a business shirt, or a combination

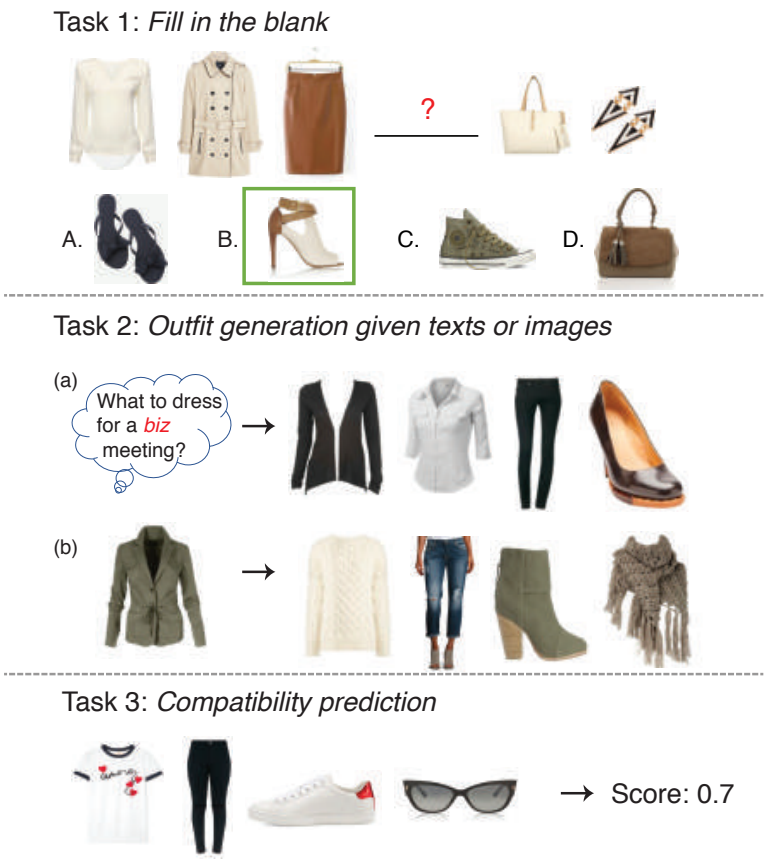


Figure 3.1: We focus on three tasks of fashion recommendation. Task 1: recommending a fashion item that matches the style of an existing set. Task 2: generating an outfit based on users' text/image inputs. Task 3: predicting the compatibility of an outfit.

of images and text, to generate a collection of fashion items for a business occasion. However, no prior approach supports multimodal inputs for recommendation.

Key to fashion recommendation is modeling the compatibility of fashion items. We contend that a compatible outfit (as shown in Figure 3.3) should have two key properties: (1) items in the outfit should be visually compatible and share similar

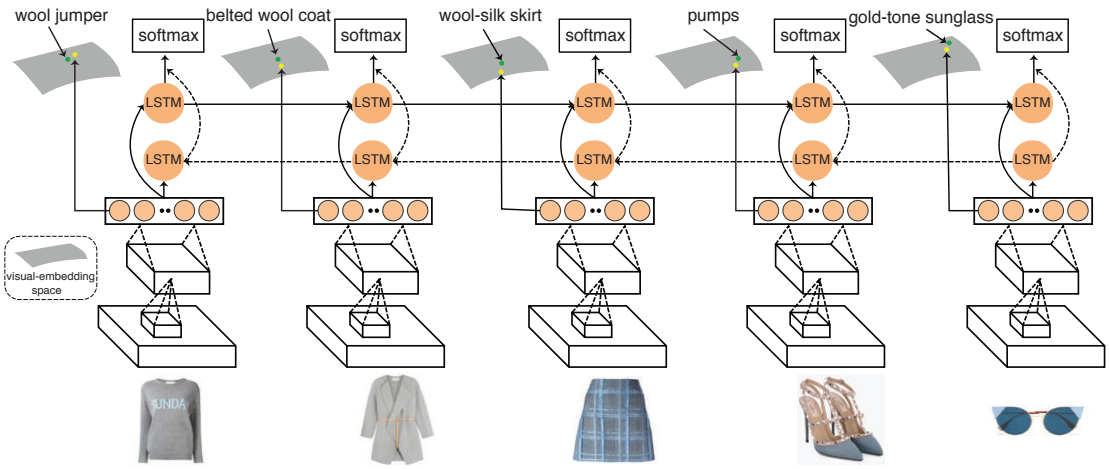


Figure 3.2: An overview of the proposed framework. We treat a given outfit as a sequence of fashion items (jumper, coat, skirt, pumps, sunglasses). Then we build a bidirectional LSTM (Bi-LSTM) to sequentially predict the next item conditioned on previously seen items in both directions. For example, given the jumper and coat, predict the skirt. Further, a visual-semantic embedding is learned by projecting images and their descriptions into a joint space to incorporate useful attribute and category information, which regularizes the Bi-LSTM and empowers recommendation with multimodal inputs.

style; (2) these items should form a complete ensemble without redundancy (*e.g.*, an outfit with only a shirt and a pair of jeans but no shoes is not compatible, neither is an outfit containing two pairs of shoes). One possible solution is to utilize semantic attributes [46], for example, "*sweat* pants" matches well with "*running* shoes". But annotating these attributes is costly and unwieldy at scale. To mitigate this issue, researchers have proposed to learn the distance between a pair of fashion items using metric learning [50] or a Siamese network [5]. However, these works estimate

pairwise compatibility relationships rather than an outfit as a whole. One could measure the compatibility of an outfit with some voting strategy using all pairs in the set, but this would incur high computational cost when the set is large and would fail to incorporate coherence among all items in the collection. On the other hand, some recent works [37, 49] attempted to predict the popularity or "fashionability" of an outfit, but they fail to handle the outfit generation task. In contrast, we are interested in modeling compatibility relationships of fashion items using their dependencies embedded in the entire outfit.

To address the above limitations, we propose to jointly learn a visual-semantic embedding and the compatibility relationships among fashion items in an end-to-end framework. Figure 3.2 gives an overview of the proposed approach. More specifically, we first adopt the Inception-V3 CNN model [51] as the feature extractor to transform an image to a feature vector. Then we utilize a one-layer bidirectional LSTM (Bi-LSTM) with 512 hidden units on top of the CNN model. Bi-LSTM [52] is a variant of Recurrent Neural Networks (RNNs) with memory cells and different functional gates governing information flow, and has have been successfully applied to temporal modeling tasks such as speech recognition [53], and image and video captioning [54, 55]. The intuition of using Bi-LSTM is that we can consider a collection of clothing items as a sequence with a specific order - top to bottom and then on to accessories (*e.g.*, shirt, pants, shoes and sunglasses) - and each image in the collection as a time step. At each time step, given the previous images, we train the Bi-LSTM model to predict the next item in the collection. Learning the transitions between time steps serves as a proxy for identifying the compatibility relationships

of fashion items. Furthermore, in addition to predicting the next image, we also learn a visual-semantic embedding by projecting the image features into a semantic representation of their descriptions. This not only provides semantic attribute and category information of the current input as a regularization for training the LSTM, but also enables the generation of an outfit with multimodal inputs from users. Finally, the model is trained end-to-end to jointly learn the compatibility relationships as well as the visual-semantic embedding.

Once the model is trained, we evaluate our network on three tasks as shown in Figure 3.1: (1) Fill-in-the-blank: given an outfit with one missing item, recommend an item that matches well with the existing set; (2) Outfit generation: generate a fashion outfit with multimodal inputs from the user; (3) Compatibility prediction: predict the compatibility of a given fashion outfit. We conduct experiments on a newly collected Polyvore dataset, and compare with state-of-the-art methods. The main contributions of this work are summarized as follows:

- We jointly learn compatibility relationships among fashion items and a visual-semantic embedding in an end-to-end framework to facilitate effective fashion recommendation in two forms.
- We employ a Bi-LSTM model to learn the compatibility relationships among fashion items by modeling an outfit as a sequence.
- Through an extensive set of experiments, we demonstrate our network outperforms several alternative methods with clear margins.

3.2 Related Work

We discuss multiple streams of works that are closely related to our approach.

Fashion Recognition and Retrieval. There is a growing interest in identifying fashion items in images due to the huge potential for commercial applications. Most recent works utilize standard segmentation methods, in combination with human pose information, to parse different garment types [56, 57] for effective retrieval. Liu *et al.* proposed a street-to-shop application that learns a mapping between photos taken by users with product images [58]. Hadi *et al.* further utilized deep learning techniques to learn the similarity between street and shop images [38]. Wang *et al.* used a robust contrastive loss to improve the retreival performance [59]. Recently, Liu *et al.* introduced FashionNet to learn fashion representations that jointly predicts clothing attributes and landmarks [19]. In contrast to these works focusing on retrieval tasks, our goal is to learn the visual compatibility relationships of fashion items in an outfit.

Fashion Recommendation. As discussed previously, there are a few approaches for recommending fashion items [46, 48, 49]. Liu *et al.* introduced an occasion-based fashion recommendation system with a latent SVM framework that relies on manually labeled attributes [46]. Hu *et al.* proposed a functional tensor factorization approach to generate an outfit by modeling the interactions between user and fashion items [48]. Recently, Li *et al.* trained an RNN to predict the popularity of a fashion set by fusing text and image features [49]. Then they constructed a recommendation by selecting the item that produces the highest popularity score when inserted into

a given set. In contrast to these approaches, our method learns the compatibility relationships among fashion items together with a visual-semantic embedding, which enables both item and outfit recommendation.

Visual Compatibility Learning. In the context of fashion analysis, visual compatibility measures whether clothing items complement one another across visual categories. For example, "sweatpants" are more compatible with "running shoes" than "high-heeled shoes". Simo-Serro *et al.* implicitly learned the compatibility of an outfit by predicting its "fashionability" [37]. McAuley *et al.* learned a distance metric between clothes with CNN features to measure their compatibilities [50]. Veit *et al.* further improved the distance metric learning with a Siamese network [5]. Recently, Oramas *et al.* mined mid-level elements to model the compatibility of clothes [60]. In this dissertation, we consider the visual compatibility of an entire outfit – items in a fashion collection are expected to share similar styles, forming a stylish composition. To this end, we leverage a Bi-LSTM model to learn the compatibility relationships for outfits, capturing the dependencies among fashion items.

Sequential Learning with LSTM. Compared with traditional RNNs, an LSTM is able to model long-range temporal dependencies across time steps without suffering the "vanishing gradients" effect. This results from the use of a memory cell regulated by different functional gates, which assist the LSTM to learn when to forget previous information and when to memorize new things. LSTM models have been successfully applied to capture temporal dependencies in sequences such as speech [53] and videos [55,61,62], *etc.* In this work, we employ an LSTM to capture the compatibility

Figure 3.3: A sample outfit from the Polyvore website. A typical outfit contains a fashion item list, *i.e.*, pairs of fashion images and their corresponding descriptions.

relationships of fashion items by considering an outfit as a sequence from top to bottom and then accessories and images in the collection as individual time steps.

3.3 Polyvore Dataset

Polyvore (www.polyvore.com) is a popular fashion website, where users create and upload outfit data as shown in Figure 3.3. These fashion outfits contain rich multimodal information like images and descriptions of fashion items, number of likes of the outfit, hash tags of the outfit, *etc.* Researchers have utilized this information for various fashion tasks [29,48,49]. However, their datasets are not publicly available.

Therefore, we collected our own dataset from Ployvore containing 21,889 outfits. These outfits are split into 17,316 for training, 1,497 for validation and 3,076 for testing. Following [49], we also use a graph segmentation algorithm to ensure there are no overlapping items between two splits. For outfits that contain too many fashion items, we only keep the first 8 for simplicity. The resulting Polyvore dataset contains 164,379 items (each item contains a pair - product image and a corresponding text description). The average number of fashion items in an outfit is 6.5. To clean the text descriptions, we remove words appearing fewer than 30 times, leading to a vocabulary of size 2,757. We choose a large threshold when filtering words because the text descriptions are very noisy and lower-ranked words have very low visualness. Note that the fashion items in an outfit on Polyvore.com are usually organized in fixed order - tops, bottoms, shoes, and the accessories. The orders of the tops and accessories are also fixed - for tops, shirts and t-shirts come before outwears; accessories are usually in the order of handbags, hats, glasses, watches, necklaces, earrings, *etc.* This enables an RNN model like an LSTM to learn "temporal" information. This dataset will be released for research purposes.

3.4 The Proposed Fashion Compatibility Modeling Approach

We next introduce the key components of the framework shown in Figure 3.2, consisting of a bidirectional LSTM for fashion compatibility modeling and a visual-semantic embedding to capture multimodal information.

3.4.1 Fashion Compatibility Learning with Bi-LSTM

The recurrent nature of LSTM models enables them to learn relationships between two time steps, and the use of memory units regulated by different cells facilitates exploiting long-term temporal dependencies. To take advantage of the representation power of LSTM, we treat an outfit as a sequence and each image in the outfit as an individual time step, and employ the LSTM to model the visual compatibility relationships of outfits.

Given a fashion image sequence $\mathbf{F} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, $\mathbf{x}_t$ is the feature representation derived from a CNN model for the t-th fashion item in the outfit. At each time step, we first use a forward LSTM to predict the next image given previous images; learning the transitions between time steps serves as a proxy for estimating the compatibility relationships among fashion items. More formally, we minimize the following objective function:

$$E_f(\mathbf{F}; \mathbf{\Theta}_f) = -\frac{1}{N}\sum_{t=1}^{N} logPr(\mathbf{x}_{t+1}|\mathbf{x}_1, ..., \mathbf{x}_t; \mathbf{\Theta}_f), \tag{3.1}$$

where $\mathbf{\Theta}_f$ denotes the model parameters of the forward prediction model and $Pr(\cdot)$, computed by the LSTM model, is the probability of seeing $\mathbf{x}_{t+1}$ conditioned on previous inputs.

More specifically, the LSTM model maps an input sequence $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ to outputs via a sequence of hidden states by computing the following equations

recursively from $t = 1$ to $t = N$:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{c}_t = \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c),$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_\mathbf{o}),$$

$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{c}_t),$$

where $\mathbf{x}_t, \mathbf{h}_t$ are the input and hidden vectors of the t-th time step, $\mathbf{i}_t, \mathbf{f}_t, \mathbf{c}_t, \mathbf{o}_t$ are the activation vectors of the input gate, forget gate, memory cell and output gate, $\mathbf{W}_{\alpha\beta}$ is the weight matrix between vector α and β (e.g., $\mathbf{W}_{xi}$ is weight matrix from the input $\mathbf{x}_t$ to the input gate $\mathbf{i}_t$), $\mathbf{b}_\alpha$ is the bias term of α and σ is the sigmoid function.

Following [63] that utilizes softmax output to predict the next word in a sentence, we append a softmax layer on top of $\mathbf{h}_t$ to calculate the probability of the next fashion item conditioned on previously seen items:

$$Pr(\mathbf{x}_{t+1}|\mathbf{x}_1, ..., \mathbf{x}_t; \mathbf{\Theta}_f) = \frac{exp(\mathbf{h}_t\mathbf{x}_{t+1})}{\sum_{\mathbf{x}\in\mathcal{X}} exp(\mathbf{h}_t\mathbf{x})}, \tag{3.2}$$

where $\mathcal{X}$ contains all images (in multiple outfits) from the current batch. This allows the model to learn discriminative style and compatibility information by looking at a diverse set of samples. Note that one can choose $\mathcal{X}$ to be the whole vocabulary [64] as in sentence generation tasks; however this is not practical during training our model due to the large number of images and high-dimensional image representations. Therefore, we set $\mathcal{X}$ to be all possible choices in the batch of $\mathbf{x}_{t+1}$

to speed up training, instead of choosing from hundreds of thousands of images from the training data.

Given a fashion item, it makes intuitive sense that predicting the next item can be performed in the reverse order also. For example, the next item for "pants" could be either "shirts" or "shoes". Therefore, we also build a backward LSTM to predict a previous item given the items after it:

$$E_b(\mathbf{F}; \mathbf{\Theta}_b) = -\frac{1}{N} \sum_{t=N-1}^{0} logPr(\mathbf{x}_t | \mathbf{x}_N, ..., \mathbf{x}_{t+1}; \mathbf{\Theta}_b), \tag{3.3}$$

and

$$Pr(\mathbf{x}_t | \mathbf{x}_N, ..., \mathbf{x}_{t+1}; \mathbf{\Theta}_b) = \frac{exp(\tilde{\mathbf{h}}_{t+1} \mathbf{x}_t)}{\sum_{\mathbf{x} \in \mathcal{X}} exp(\tilde{\mathbf{h}}_{t+1} \mathbf{x})}, \tag{3.4}$$

where $\tilde{\mathbf{h}}_{t+1}$ is the hidden state at time $t+1$ of the backward LSTM, and $\mathbf{\Theta}_b$ denotes the backward prediction model parameters. Note that we add two zero vectors $\mathbf{x}_0$ and $\mathbf{x}_{N+1}$ in $\mathbf{F}$ so that the bidirectional LSTM learns when to stop predicting the next item.

Since an outfit is usually a stylish ensemble of fashion items that share similar styles (*e.g.*, color or texture), by treating an outfit as an ordered sequence, the Bi-LSTM model is trained explicitly to capture compatibility relationships as well as the overall style of the entire outfit (knowledge learned in the memory cell). This makes it a very good fit for fashion recommendation.

3.4.2 Visual-semantic Embedding

Fashion recommendation should naturally be based on multimodal inputs (exemplar images and text describing certain attributes) from users. Therefore, it is important to learn a multimodal embedding space of texts and images. Instead of annotating images with labels or attributes, which is costly, we leverage the weakly-labeled web data, *i.e.*, the informative text description of each image provided by the dataset, to capture multimodal information. To this end, we train a visual-semantic embedding by projecting images and their associated text into a joint space, which is widely used when modeling image-text pairs [23].

Given a fashion image from an outfit, its description is denoted as $S = \{w_1, w_2, ..., w_M\}$ where w_i represents each word in the description. We first represent the i-th word w_i with one-hot vector $\mathbf{e}_i$, and transform it into the embedding space by $\mathbf{v}_i = \mathbf{W_T} \cdot \mathbf{e}_i$ where $\mathbf{W_T}$ represents the word embedding matrix. We then encode the description with bag-of-words $\mathbf{v} = \frac{1}{M} \sum_i \mathbf{v}_i$. Letting $\mathbf{W_I}$ denote the image embedding matrix, we project the image representation $\mathbf{x}$ into the embedding space and represent it as $\mathbf{f} = \mathbf{W_I} \cdot \mathbf{x}$.

In the visual-semantic space, we estimate the similarity between an image and its description with their cosine distance: $d(\mathbf{f}, \mathbf{v}) = \mathbf{f} \cdot \mathbf{v}$, where $\mathbf{f}$ and $\mathbf{v}$ are normalized to unit norm. Finally, the images and descriptions are embedded in the joint space by minimizing the following contrastive loss:

$$E_e(\mathbf{\Theta}_e) = \sum_{\mathbf{f}} \sum_{k} \max(0, m - d(\mathbf{f}, \mathbf{v}) + d(\mathbf{f}, \mathbf{v}_k)) + \sum_{\mathbf{v}} \sum_{k} \max(0, m - d(\mathbf{v}, \mathbf{f}) + d(\mathbf{v}, \mathbf{f}_k)), \tag{3.5}$$

where $\mathbf{\Theta}_e = \{\mathbf{W_I}, \mathbf{W_T}\}$ are the model parameters, and $\mathbf{v}_k$ denotes non-matching descriptions for image $\mathbf{f}$ while $\mathbf{f}_k$ are non-matching images for description $\mathbf{v}$. By minimizing this loss function, the distance between $\mathbf{f}$ and its corresponding description $\mathbf{v}$ is forced to be smaller than the distance from unmatched descriptions $\mathbf{v}_k$ by some margin m. Vice versa for description $\mathbf{v}$. During the training, all non-matching pairs inside each mini batch are selected to optimize Eqn. 3.5. As such, fashion items that share similar semantic attributes and styles will be close in the learned embedding space.

3.4.3 Joint Modeling

Given a fashion output, the Bi-LSTM is trained to predict the next or previous item by utilizing the visual compatibility relationships. However, this is not optimal since it overlooks the semantic information and also prevents users from using multimodal input to generate outfits. Therefore, we propose to jointly learn fashion compatibility and the visual-semantic embedding with an aim to incorporate semantic information in the training process of the Bi-LSTM. The overall objective function is described as follows:

$$\min_{\mathbf{\Theta}} \quad \sum_{\mathbf{F}} (E_f(\mathbf{F}; \mathbf{\Theta}_f) + E_b(\mathbf{F}; \mathbf{\Theta}_b)) + E_e(\mathbf{\Theta}_e), \tag{3.6}$$

where $\mathbf{\Theta} = \{\mathbf{\Theta}_f, \mathbf{\Theta}_b, \mathbf{\Theta}_e\}$. The first two terms in Eqn. 3.6 are the Bi-LSTM objective functions, and the third term computes the visual-semantic embedding loss. The framework can be easily trained by Back-Propagation through time (BPTT) [52] in an end-to-end fashion, in which gradients are aggregated through time. The only difference compared to a standard Bi-LSTM model during back-propagation is that the gradients of the CNN model now stem from the average of two sources (See Figure 3.2), allowing the CNN model to learn useful semantic information at the same time. The visual-semantic embedding not only serves as a regularization for the training of Bi-LSTM but also enables multimodal fashion recommendation as will be demonstrated in the next section.

3.5 Experiment

In this section, we first introduce the experiment settings. Then we conduct an extensive set of experiments to validate the effectiveness of the proposed approach on three tasks, including fill-in-the-blank fashion recommendation (Sec. 3.5.3), fashion compatibility prediction (Sec. 3.5.4) and fashion outfit generation (Sec. 3.5.5).

3.5.1 Implementation Details

Bidirectional LSTM. We use 2048D CNN features derived from the GoogleNet InceptionV3 model [51] as the image representation, and transform the features into 512D with one fully connected layer before feeding them into the Bi-LSTM. The number of hidden units of the LSTM is 512, and we set the dropout rate to 0.7.

Visual-semantic Embedding. The dimension of the joint embedding space is set to 512, and thus $\mathbf{W_I} \in \mathbb{R}^{2048\times512}$ and $\mathbf{W_T} \in \mathbb{R}^{2757\times512}$, where 2757 is the size of the vocabulary. We fix the margin $m = 0.2$ in Eqn. 3.5.

Joint Training. The initial learning rate is 0.2 and is decayed by a factor of 2 every 2 epochs. The batch size is set to 10, and thus each mini batch contains 10 fashion outfit sequences, around 65 images and their corresponding descriptions. Finally, we fine-tune all layers of the network pre-trained on ImageNet. We stop the training process when the loss on the validation set stabilizes.

3.5.2 Compared Approaches

To demonstrate the effectiveness of our approach for modeling the compatibility of fashion outfits, we compare with the following alternative methods:

SiameseNet [5]. SiameseNet utilizes a Siamese CNN to project two clothing items into a latent space to estimate their compatibility. To compare with SiameseNet, we train a network with the same structure by considering fashion items in the same outfit as positive compatible pairs and items from two different outfits as negative pairs. The compatibility of an outfit is obtained by averaging pairwise compatibility, in the form of cosine distance in the learned embedding, of all pairs in the collection. For fair comparisons, the embedding size is also set to 512. We also normalize the embedding with ℓ_2 norm before calculating the Siamese loss, and set the margin parameter to 0.8.

SetRNN [49]. Given a sequence of fashion images, SetRNN predicts the fashion

set popularity using an RNN model. We use the popularity prediction of SetRNN as the set compatibility score.

Visual-semantic Embedding (VSE). We only learn a VSE by minimizing E_e in Eqn. 3.5 without training any LSTM model. The resulting embeddings are used to measure the compatibility of an outfit, similar to SiameseNet.

Bi-LSTM. Only a bidirectional LSTM is trained without incorporating any semantic information.

F-LSTM+VSE. Jointly training the forward LSTM with visual-semantic embedding, *i.e.*, minimizing $E_f + E_e$.

B-LSTM+VSE. Similarly, only a backward LSTM is trained with visual-semantic embedding, *i.e.* minimizing $E_b + E_e$.

Bi-LSTM+VSE. Our full model by jointly learning the bidirectional LSTM and the visual-semantic embedding.

The first two approaches are recent works in this line of research and the remaining methods are used for ablation studies to analyze the contribution of each component in our proposed framework. The hyper-parameters in these methods are chosen using the validation set.

3.5.3 Fill-in-the-blank Fashion Recommendation

Recently, several fill-in-the-blank (FITB) datasets [65–68] have been created and evaluated to bridge visual and semantic information. However, no existing dataset deals with image sequence completion (*i.e.*, given a sequence of images and

a blank, fill in the blank with a suitable image). Thus, we introduce the problem of filling-in-the-blank questions from multiple choices as shown in Task 1 of Figure 3.1. In this task, a sequence of fashion items are provided and one needs to choose an item from multiple choices that is compatible with other items to fill in the blank. This is a very practical scenario in real life, *e.g.*, a user wants to choose a pair of shoes to match his pants and coat.

To this end, we create a fill-in-the-blank dataset using all outfits in the Polyvore test set. For each outfit, we randomly select one item and replace it with a blank, and then select 3 items from other outfits along with the ground truth item to obtain a multiple choice set. We believe that a randomly selected item should be less compatible than the one chosen by experienced designers on Polyvore. Thus, it is reasonable to evaluate fashion recommendation methods on such multiple-choice questions. Once our Bi-LSTM+VSE is trained, we solve the fill-in-the-blank task based on the following objective function:

$$\mathbf{x}_a = \arg\max_{\mathbf{x}_c \in \mathcal{C}} Pr(\mathbf{x}_c|\mathbf{x}_1, ..., \mathbf{x}_{t-1}) + Pr(\mathbf{x}_c|\mathbf{x}_N, ..., \mathbf{x}_{t+1}) \tag{3.7}$$

$$= \arg\max_{\mathbf{x}_c \in \mathcal{C}} \frac{exp(\mathbf{h}_{t-1}\mathbf{x}_c)}{\sum_{\mathbf{x}\in\mathcal{C}} exp(\mathbf{h}_{t-1}\mathbf{x})} + \frac{exp(\tilde{\mathbf{h}}_{t+1}\mathbf{x}_c)}{\sum_{\mathbf{x}\in\mathcal{C}} exp(\tilde{\mathbf{h}}_{t+1}\mathbf{x})} \tag{3.8}$$

where $\mathcal{C}$ is the choice set, and t is the position of the blank we aim to fill in. Hence, during inference time, forward and backward LSTMs independently predict the probability of one candidate belonging to the outfit, and the candidate having the highest total probability is selected as the answer.

The middle column of Table 3.1 shows the results of our method compared

Figure 3.4: Examples of our method on the fill-in-the-blank task. Green bounding boxes indicate the correct answers, while red box shows a failure case. Prediction score of each choice is also displayed.

Method	FITB accuracy	Compatibility AUC
SetRNN [49]	29.6%	0.53
SiameseNet [5]	52.0%	0.85
VSE	29.2%	0.56
F-LSTM + VSE	63.7%	0.89
B-LSTM + VSE	61.2%	0.88
Bi-LSTM	66.7%	0.89
Bi-RNN + VSE	63.7%	0.85
Bi-GRU + VSE	67.1%	0.89
Bi-LSTM + VSE (Ours)	**68.6%**	**0.90**

Table 3.1: Comparison between our method and other methods on the fill-in-the-blank (FITB) and compatibility prediction tasks.

with alternative approaches on this task. From this table, we make the following observations: 1) SetRNN and VSE perform similar to random guess (25%); thus they are not suitable for this task. SetRNN predicts popularity of an outfit, but popularity does not always indicate good compatibility. Similar retrieval accuracy is also observed in the SetRNN paper [49]. VSE does not work very well due to the noises in text labels, and also its failure to model the relationships of items in one outfit. 2) SiameseNet works better than VSE and SetRNN but still worse than LSTM based methods, since it mainly considers pairwise relationships rather than the compatibility of the entire outfit; thus it sometimes chooses a candidate with a

category that is already in the outfit though the styles are indeed similar. 3) F-LSTM outperforms B-LSTM. We attribute this to the fact that the last several items in most of the outfits are accessories, and it is harder for the backward LSTM to predict clothing items based on accessories than the other way around. The combination of LSTMs in these two directions offers higher accuracy than one directional LSTM. 4) We further jointly learn the Bi-LSTM with the visual-semantic embedding, and the resulting full model achieves the best performance with an accuracy of 68.6%, 1.9 percentage points higher than Bi-LSTM alone. This verifies the assumption the visual-semantic embedding can indeed assist the training of Bi-LSTM by providing semantic clues like classes and attributes. 5) We also investigate different RNN architectures by replacing LSTM cells with gated recurrent unit (GRU) and basic RNN cells. GRU and LSTM are better than basic RNN by better addressing the "vanishing gradients" effect and better modeling the temporal dependencies. The choice between LSTM and GRU depends heavily on the dataset and corresponding task [69]; our experiments demonstrate that LSTM is more suitable for modeling compatibility of fashion items.

In Figure 3.4, we visualize sample results of our method for the filling-in-the-blank task. Combining Bi-LSTM and visual-semantic embedding can not only detect what kinds of fashion item is missing (*e.g.*, coat is missing in the second example of the Figure 3.4), but also selects the fashion item that is most compatible to the query items and matches their style as well (*e.g.*, running shoes are more compatible with the sporty outfit in the third example of Figure 3.4).

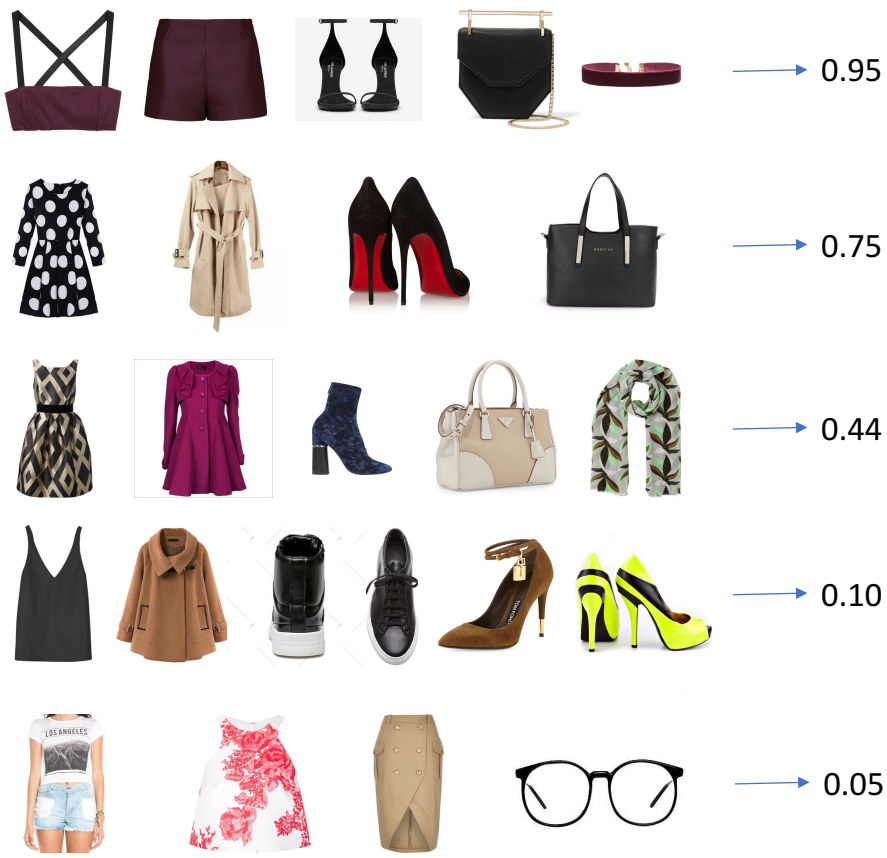


Figure 3.5: Results of our method on the fashion outfit compatibility prediction task. Scores are normalized to be between 0 and 1 for better visualization.

3.5.4 Fashion Compatibility Prediction

In addition to recommending fashion items, our model can also predict the compatibility of an outfit. This is useful since users may create their own outfits and wish to determine if they are compatible and trendy. Even though minimizing Eqn. 3.6 does not explicitly predict compatibility, since our model is trained on the outfit data generated on Polyvore which are usually fashionable and liked by a lot of users, it can be used for this purpose. Given an outfit $\mathbf{F}$, we simply utilize the

value of the first two terms in Eqn. 3.6 (Bi-LSTM prediction loss) as an indicator of compatibility.

To compare with alternative methods, similarly to the filling-in-the-blank dataset, we created 4,000 *incompatible* outfits by randomly selecting fashion items from the test set. The performance is evaluated using the AUC of the ROC curve. Results are presented in the third column of Table 3.1. Our method obtains the best performance among all methods, outperforming recent works [5, 49] by clear margins. Particularly, it is interesting to see that our method, designed to learn the compatibility relationships by predicting the next item conditioned on previous items, is significantly better than SetRNN, which is directly trained to predict set popularity. In addition, we also observe that one directional LSTM is good enough for compatibility prediction.

Figure 3.5 shows qualitative results of our method. From this figure, we can observe that our method can predict if a set of fashion items forms a compatible (stylish) outfit. For example, the outfit in the first row contains purple/black items with the same style and thus has a high compatibility score; all the items in the third row have different colors, which makes them somewhat incompatible to form an outfit; the fourth outfit contains 4 pairs of shoes without a bottom, and the last one contains two dresses but no shoes; thus they are both incompatible outfits.

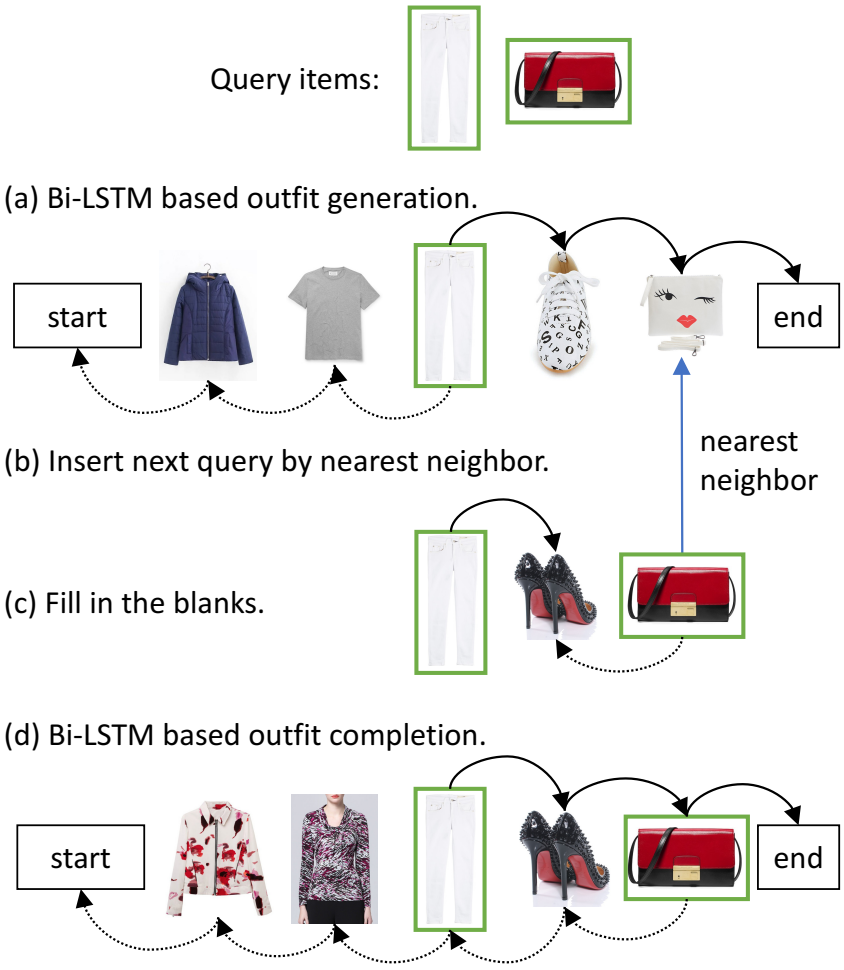


Figure 3.6: Given query fashion images, our method can generate a compatible outfit.

3.5.5 Fashion Outfit Generation

We now discuss how to utilize our proposed framework to generate an outfit with multimodal specifications (images/text) from users.

Generate Outfits from Query Images. Figure 3.6 gives an overview of this process. We first consider a degenerate scenario where users provide a single image and wish to obtain an entire outfit with consistent style. This can be accomplished

Figure 3.7: Fashion outfit recommendation given query items. Each row contains a recommended outfit where query images are indicated by green boxes.

simply by running the trained Bi-LSTM in two directions as shown in Figure 3.6 (a). When users provide more than one item, we first utilize the first item to generate an initial outfit and then find and replace the nearest neighbor of the next query item (Figure 3.6 (b)). If the two items are contiguous, we can perform inference in both directions to produce an outfit. Otherwise, we fill in all the blanks between these two items to achieve coherence before performing inference (Figure 3.6 (c)). This ensures the subsequence used to generate the entire outfit is visually compatible. When more input images are available, this process can be repeated recursively. Finally, the outfit is generated by running the Bi-LSTM model in both directions on the subsequence (Figure 3.6 (d)). We can see that many fashion items are visually compatible with the white pants, and the initial outfit generated in Figure 3.6 (a) has a casual style. When incorporating the black/red handbag, our model first predicts a pair of black/red shoes that match both items, and automatically generates an outfit with a slightly more formal style accordingly.

We demonstrate sample outfit generation results given one to three image inputs in Figure 3.7. It is clear that our method can produce visually compatible and complete outfits. Note that we only show qualitative results of our method since SiameseNet [5], SetRNN [49] and VSE cannot tackle this task.

Generate Outfits from Multimodal Queries. Since we jointly learn a visual-semantic embedding together with the Bi-LSTM, our method can also take an auxiliary text query and generate an outfit that is not only visually compatible with the given query fashion items, but also semantically relevant to the given text query. This can be done by first generating an initial outfit using Bi-LSTM based on the

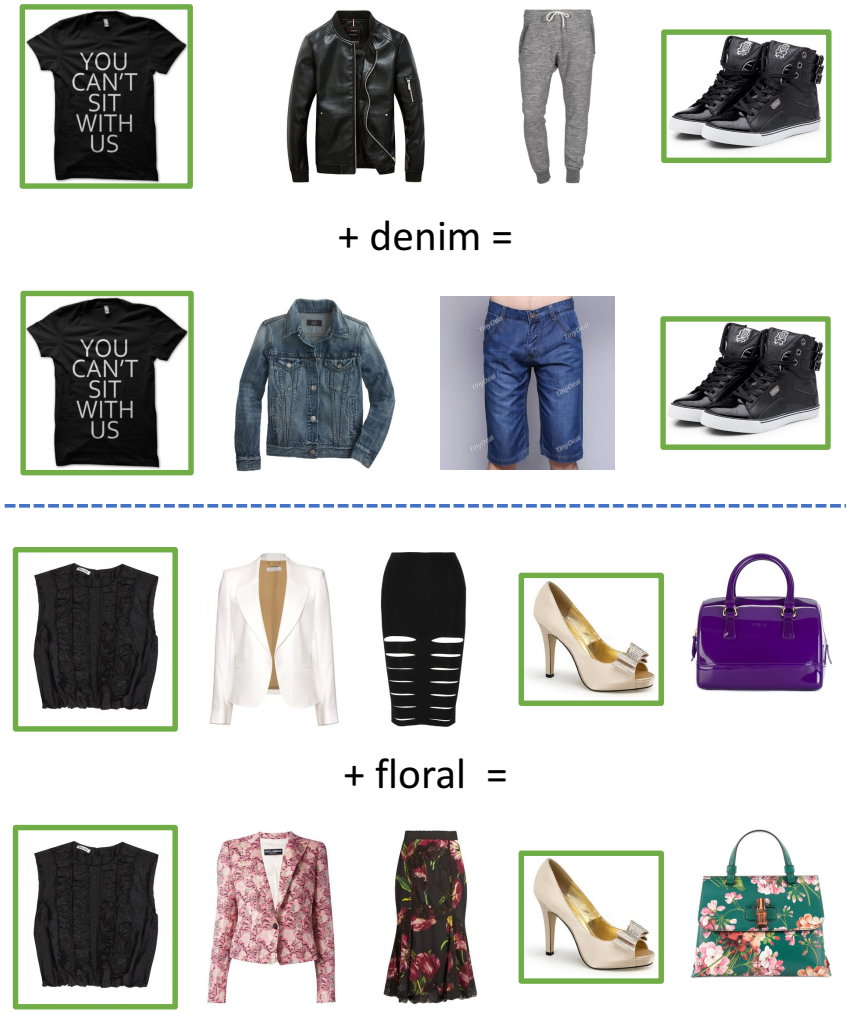


Figure 3.8: Fashion outfit recommendation given query items and text input. Query images are indicated by green boxes. Outfits on the top are generated without using the text input. When a text query is provided the outfits are adjusted accordingly.

given fashion items. Then, given the semantic representation of the text query $\mathbf{v}_q$, each non-query item $\mathbf{f}_i$ in the initial outfit is updated by $\arg\min_{\mathbf{f}} d(\mathbf{f}, \mathbf{f}_i + \mathbf{v}_q)$. Thus, the updated item is both similar to the original item and also close to the text query in the visual-semantic embedding space. Figure 3.8 shows two examples of our recommended fashion outfits when multimodal queries are provided. Our model

Figure 3.9: Fashion outfit recommendation given text input. The input can either be an attribute or style (*e.g.*, *denim*, *casual*) or descriptions of fashion items (*e.g.*, *lace dress* + *red pump*).

effectively generates visually compatible and semantically relevant outfits.

Generate Outfits from Text Queries. In addition to combining images and text inputs, our model is also capable of generating outfits given only text inputs. We can take two kinds of text inputs from users - an attribute or style that all items are expected to share, or descriptions of items the generated outfit should contain. In the first scenario, the nearest image to the text query is chosen as the query image, and then the Bi-LSTM model can produce an outfit using this image. Then, the

outfit is updated in the same manner as when both image and text inputs are given (the first two examples in Figure 3.9). In the other scenario, a fashion item image is retrieved using each description, and all images are treated as query images to generate the outfit (the last two examples in Figure 3.9).

3.6 Conclusion

In this part of dissertation, we propose to jointly train a Bi-LSTM model and a visual-semantic embedding for fashion compatibility learning. We consider an outfit as a sequence and each item in the outfit as an time step, and we utilize a Bi-LSTM model to predict the next item conditioned on previously seen ones. We also train a visual-semantic embedding to provide category and attribute information in the training process of the Bi-LSTM. We conducted experiments on different types of fashion recommendation tasks using our newly collected Polyvore dataset, and the results demonstrate that our method can effectively learn the compatibility of fashion outfits. Since fashion compatibility might vary from one person to another, modeling user-specific compatibility and style preferences is one of our future research directions.

Chapter 4: VITON: An Image-based Virtual Try-on Network

4.1 Introduction

Recent years have witnessed the increasing demands of online shopping for fashion items. Online apparel and accessories sales in US are expected to reach 123 billion in 2022 from 72 billion in 2016 [4]. Despite the convenience online fashion shopping provides, consumers are concerned about how a particular fashion item in a product image would look on them when buying apparel online. Thus, allowing consumers to virtually try on clothes will not only enhance their shopping experience, transforming the way people shop for clothes, but also save cost for retailers. Motivated by this, various virtual fitting rooms/mirrors have been developed by different companies such as TriMirror, Fits Me, *etc.* However, the key enabling factor behind them is the use of 3D measurements of body shape, either captured directly by depth cameras [70] or inferred from a 2D image using training data [71,72]. While these 3D modeling techniques enable realistic clothing simulations on the person, the high costs of installing hardwares and collecting 3D annotated data inhibit their large-scale deployment.

We present an image-based virtual try-on approach, relying merely on plain RGB images *without leveraging any 3D information*. Our goal is to synthesize a

Figure 4.1: Virtual try-on results generated by our method. Each row shows a person virtually trying on different clothing items. Our model naturally renders the items onto a person while retaining her pose and preserving detailed characteristics of the target clothing items.

photo-realistic new image by overlaying a product image seamlessly onto the corresponding region of a clothed person (as shown in Figure 4.1). The synthetic image is expected to be perceptually convincing, meeting the following desiderata: (1) body parts and pose of the person are the same as in the original image; (2) the clothing

item in the product image deforms naturally, conditioned on the pose and body shape of the person; (3) detailed visual patterns of the desired product are clearly visible, which include not only low-level features like color and texture but also complicated graphics like embroidery, logo, *etc.* The non-rigid nature of clothes, which are frequently subject to deformations and occlusions, poses a significant challenge to satisfying these requirements simultaneously, especially without 3D information.

Conditional Generative Adversarial Networks (GANs), which have demonstrated impressive results on image generation [73, 74], image-to-image translation [75] and editing tasks [76], seem to be a natural approach for addressing this problem. In particular, they minimize an adversarial loss so that samples generated from a generator are indistinguishable from real ones as determined by a discriminator, conditioned on an input signal [73,75,77,78]. However, they can only transform information like object classes and attributes roughly, but are unable to generate graphic details and accommodate geometric changes [79]. This limits their ability in tasks like virtual try-on, where visual details and realistic deformations of the target clothing item are required in generated samples.

To address these limitations, we propose a virtual try-on network (VITON), a coarse-to-fine framework that seamlessly transfers a target clothing item in a product image to the corresponding region of a clothed person in a 2D image. Figure 4.2 gives an overview of VITON. In particular, we first introduce a clothing-agnostic representation consisting of a comprehensive set of features to describe different characteristics of a person. Conditioned on this representation, we employ a multi-task encoder-decoder network to generate a coarse synthetic clothed person in the

same pose wearing the target clothing item, and a corresponding clothing region mask. The mask is then used as a guidance to warp the target clothing item to account for deformations. Furthermore, we utilize a refinement network which is trained to learn how to composite the warped clothing item to the coarse image so that the desired item is transfered with natural deformations and detailed visual patterns. To validate our approach, we conduct a user study on our newly collected dataset and the results demonstrate that VITON generates more realistic and appealing virtual try-on results outperforming state-of-the-art methods.

4.2 Related Work

Fashion analysis. Extensive studies have been conducted on fashion analysis due to its huge profit potentials. Most existing methods focus on clothing parsing [45,57], clothing recognition by attributes [19], matching clothing seen on the street to online products [38, 58], fashion recommendation [48], visual compatibility learning [5, 36], and fashion trend prediction [80]. Compared to these lines of work, we focus on virtual try-on with only 2D images as input. Our task is also more challenging compared to recent work on interactive search that simply modifies attributes (*e.g.*, color and textures) of a clothing item [20, 47, 81], since virtual try-on requires preserving the details of a target clothing image as much as possible, including exactly the same style, embroidery, logo, text, *etc.*

Image synthesis. GANs [82] are one of most popular deep generative models for image synthesis, and have demonstrated promising results in tasks like image

generation [83, 84] and image editing [76, 85]. To incorporate desired properties in generated samples, researchers also utilize different signals, in the form of class labels [77], text [73], attributes [86], *etc.*, as priors to condition the image generation process. There are a few recent studies investigating the problem of image-to-image translation using conditional GANs [75], which transform a given input image to another one with a different representation. For example, producing an RGB image from its corresponding edge map, semantic label map, *etc.*, or *vice versa*. Recently, Chen and Kolton [87] trained a CNN using a regression loss as an alternative to GANs for this task without adversarial training. These methods are able to produce photo-realistic images, but have limited success when geometric changes occur [79]. Instead, we propose a refinement network that pays attention to clothing regions and deals with clothing deformations for virtual try-on.

In the context of image synthesis for fashion applications, Yoo *et al.* [88] generated a clothed person conditioned on a product image and *vice versa* regardless of the person's pose. Lassner *et al.* [74] described a generative model of people in clothing, but it is not clear how to control the fashion items in the generated results. A more related work is FashionGAN [89], which replaced a fashion item on a person with a new one specified by text descriptions. In contrast, we are interested in the precise replacement of the clothing item in a reference image with a target item, and address this problem with a novel coarse-to-fine framework.

Virtual try-on. There is a large body of work on virtual try-on, mostly conducted in computer graphics. Guan *et al.* proposed DRAPE [90] to simulate 2D clothing designs on 3D bodies in different shapes and poses. Hilsmann and P. Eisert [91]

retextured the garment dynamically based on a motion model for real-time visualization in a virtual mirror environment. Sekine *et al.* [70] introduced a virtual fitting system that adjusts 2D clothing images to users through inferring their body shapes with depth images. Recently, Pons-Moll *et al.* [92] utilized a multi-part 3D model of clothed bodies for clothing capture and retargeting. Yang *et al.* [72] recovered a 3D mesh of the garment from a single view 2D image, which is further re-targeted to other human bodies. In contrast to relying on 3D measurements to perform precise clothes simulation, in our work, we focus on synthesizing a perceptually correct photo-realistic image directly from 2D images, which is more computationally efficient. In computer vision, limited work has explored the task of virtual try-on. Recently, Jetchev and Bergmann [93] proposed a conditional analogy GAN to swap fashion articles. However, during testing, they require the product images of both the target item and the original item on the person, which makes it infeasible in practical scenarios. Moreover, without injecting any person representation or explicitly considering deformations, it fails to generate photo-realistic virtual try-on results.

4.3 Virtual Try-on Network

The goal of VITON is, given a reference image I with a clothed person and a target clothing item c, to synthesize a new image $\hat{I}$, where c is transferred naturally onto the corresponding region of the same person whose body parts and pose information are preserved. Key to a high-quality synthesis is to learn a proper trans-

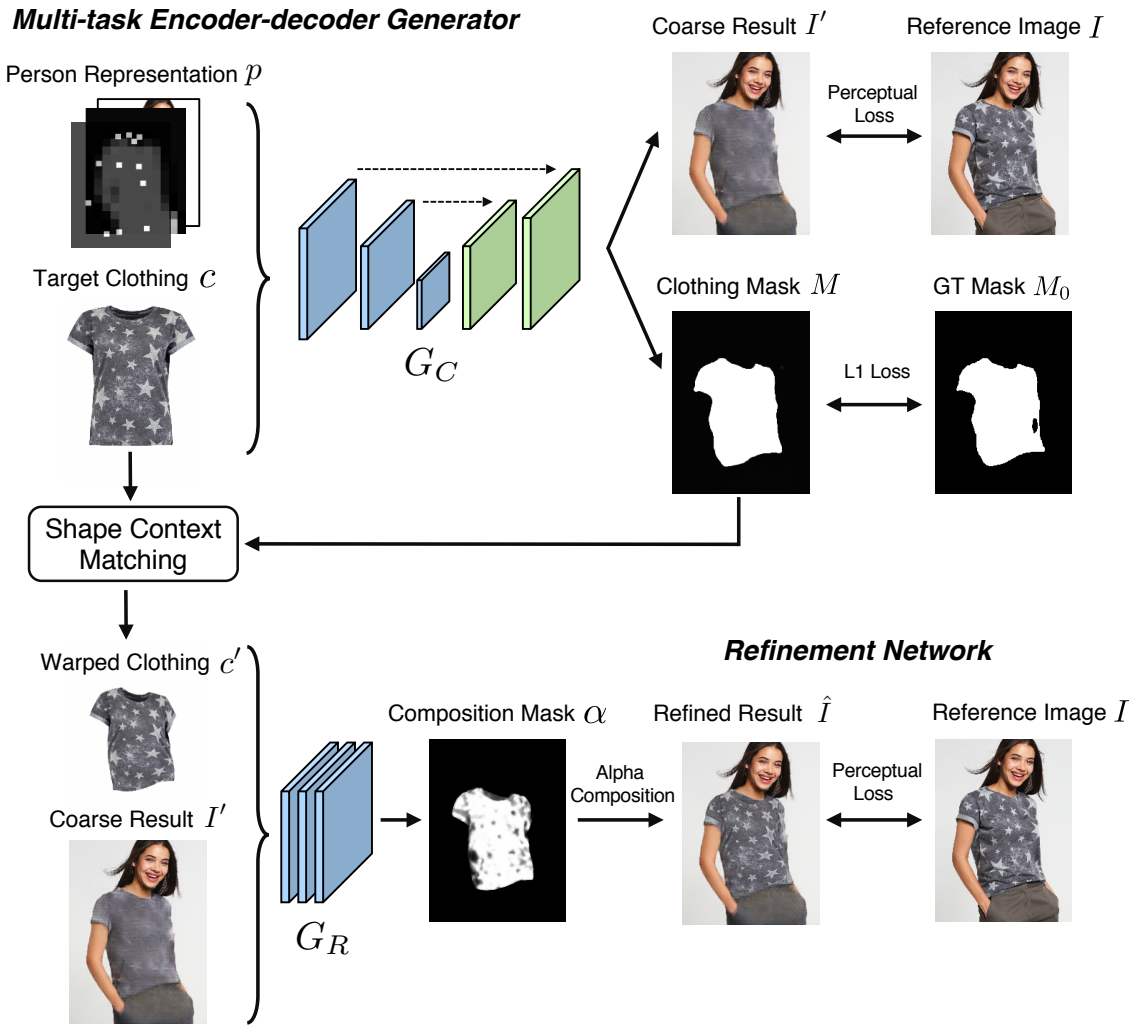


Figure 4.2: An overview of VITON. VITON consists of two stages: (a) an encoder-decoder generator stage (Sec 4.3.2), and (b) a refinement stage (Sec 4.3.3).

formation from product images to clothes on the body. A straightforward approach is to leverage training data of a person with fixed pose wearing different clothes and the corresponding product images, which, however, is usually difficult to acquire.

In a practical virtual try-on scenario, only a reference image and a desired product image are available at test time. Therefore, we adopt the same setting for training, where a reference image I with a person wearing c and the product image of c are given as inputs (we will use c to refer to the product image of c in the following sections). Now the problem becomes given the product image c and the person's information, how to learn a generator that not only produces I during training, but more importantly is able to generalize at test time – synthesizing a perceptually convincing image with an arbitrary desired clothing item.

To this end, we first introduce a clothing-agnostic person representation (Sec 4.3.1). We then synthesize the reference image with an encoder-decoder architecture (Sec 4.3.2), conditioned on the person representation as well as the target clothing image. The resulting coarse result is further improved to account for detailed visual patterns and deformations with a refinement network (Sec 4.3.3). The overall framework is illustrated in Figure 4.2.

4.3.1 Person Representation

A main technical challenge of a virtual try-on synthesis is to deform the target clothing image to fit the pose of a person. To this end, we introduce a clothing-agnostic person representation, which contains a set of features (Figure 4.3), includ-

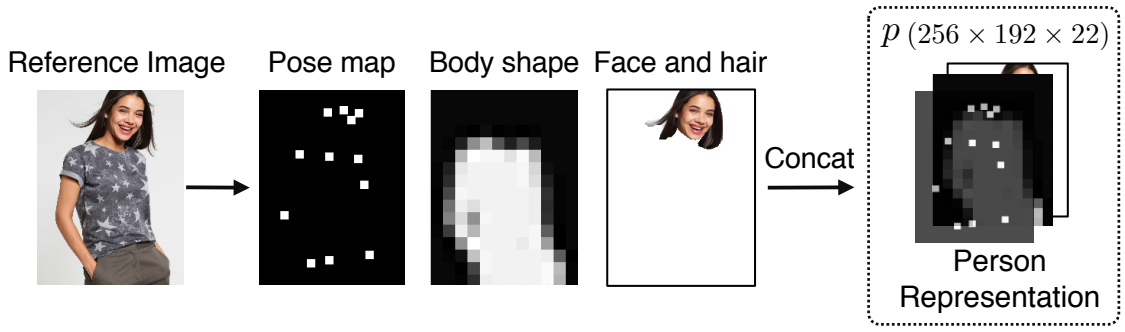


Figure 4.3: A clothing-agnostic person representation. Given a reference image I, we extract the pose, body shape and face and hair regions of the person, and use this information as part of input to our generator.

ing pose, body parts, face and hair, as a prior to constrain the synthesis process.

Pose heatmap. Variations in human poses lead to different deformations of clothing, and hence we explicitly model pose information with a state-of-the-art pose estimator [94]. The computed pose of a person is represented as coordinates of 18 keypoints. To leverage their spatial layout, each keypoint is further transformed to a heatmap, with an 11×11 neighborhood around the keypoint filled in with ones and zeros elsewhere. The heatmaps from all keypoints are further stacked into an 18-channel pose heatmap.

Human body representation. The appearance of clothing highly depends on body shapes, and thus how to transfer the target fashion item depends on the location of different body parts (*e.g.*, arms or torso) and the body shape. A state-of-the-art human parser [95] is thus used to compute a human segmentation map, where different regions represent different parts of human body like arms, legs, *etc.* We further convert the segmentation map to a 1-channel binary mask, where ones

indicate human body (except for face and hair) and zeros elsewhere. This binary mask derived directly from I is downsampled to a lower resolution (16×12 as shown in Figure 4.3) to avoid the artifacts when the body shape and target clothing conflict as in [89].

Face and hair segment. To maintain the identity of the person, we incorporate physical attributes like face, skin color, hair style, *etc.* We use the human parser [95] to extract the RGB channels of face and hair regions of the person to inject identity information when generating new images.

Finally, we resize these three feature maps to the same resolution and then concatenate them to form a clothing-agnostic person representation p such that $p \in \mathrm{R}^{\mathrm{m}\times\mathrm{n}\times\mathrm{k}}$, where $m = 256$ and $n = 192$ denote the height and width of the feature map, and $k = 18 + 1 + 3 = 22$ represents the number of channels. The representation contains abundant information about the person upon which convolutions are performed to model their relations. Note that our representation is more detailed than previous work [78, 89].

4.3.2 Multi-task Encoder-Decoder Generator

Given the clothing-agnostic person representation p and the target clothing image c, we propose to synthesize the reference image I through reconstruction such that a natural transfer from c to the corresponding region of p can be learned. In particular, we utilize a multi-task encoder-decoder framework that generates a clothed person image along with a clothing mask of the person as well. In addition to

guiding the network to focus on the clothing region, the predicted clothing mask will be further utilized to refine the generated result, as will be discussed in Sec 4.3.3. The encoder-decoder is a general type of U-net architecture [96] with skip connections to directly share information between layers through bypassing connections.

Formally, let G_C denote the function approximated by the encoder-decoder generator. It takes the concatenated c and p as its input and generates a 4-channel output $(I', M) = G_C(c, p)$, where the first 3 channels represent a synthesized image I' and the last channel M represents a segmentation mask of the clothing region as shown at the top of Figure 4.2. We wish to learn a generator such that I' is close to the reference image I and M is close to M_0 (M_0 is the pseudo ground truth clothing mask predicted by the human parser on I). A simple way to achieve this is to train the network with an L_1 loss, which generates decent results when the target is a binary mask like M_0. However, when the desired output is a colored image, L_1 loss tends to produce blurry images [75]. Following [97–99], we utilize a perceptual loss that models the distance between the corresponding feature maps of the synthesized image and the ground truth image, computed by a visual perception network. The loss function of the encoder-decoder can now be written as the sum of a perceptual loss and an L_1 loss:

$$L_{G_C} = \sum_{i=0}^{5} \lambda_i ||\phi_i(I') - \phi_i(I)||_1 + ||M - M_0||_1, \tag{4.1}$$

where $\phi_i(y)$ in the first term is the feature map of image y of the i-th layer in the visual perception network ϕ, which is a VGG19 [100] network pre-trained on ImageNet. For layers $i \geqslant 1$, we utilize 'conv1_2', 'conv2_2', 'conv3_2', 'conv4_2',

'conv5_2' of the VGG model while for layer 0, we directly use RGB pixel values. The hyperparameter λ_i controls the contribution of the i-th layer to the total loss. The perceptual loss forces the synthesized image to match RGB values of the ground truth image and their activations at different layers in a visual perception model as well, allowing the synthesis network to learn realistic patterns. The second term in Eqn. 4.1 is a regression loss that encourages the predicted clothing mask M to be the same as M_0.

By minimizing Eqn. 4.1, the encoder-decoder learns how to transfer the target clothing conditioned on the person representation. While the synthetic clothed person conforms to the pose, body parts and identity in the original image (as illustrated in the third column of Figure 4.5), details of the target item such as text, logo, *etc.* are missing. This might be attributed to the limited ability to control the process of synthesis in current state-of-the-art generators. They are typically optimized to synthesize images that look similar globally to the ground truth images without knowing where and how to generate details. To address this issue, VITON uses a refinement network together with the predicted clothing mask M to improve the coarse result I'.

4.3.3 Refinement Network

The refinement network G_R in VITON is trained to render the coarse blurry region leveraging realistic details from a deformed target item.

Warped clothing item. We borrow information directly from the target clothing

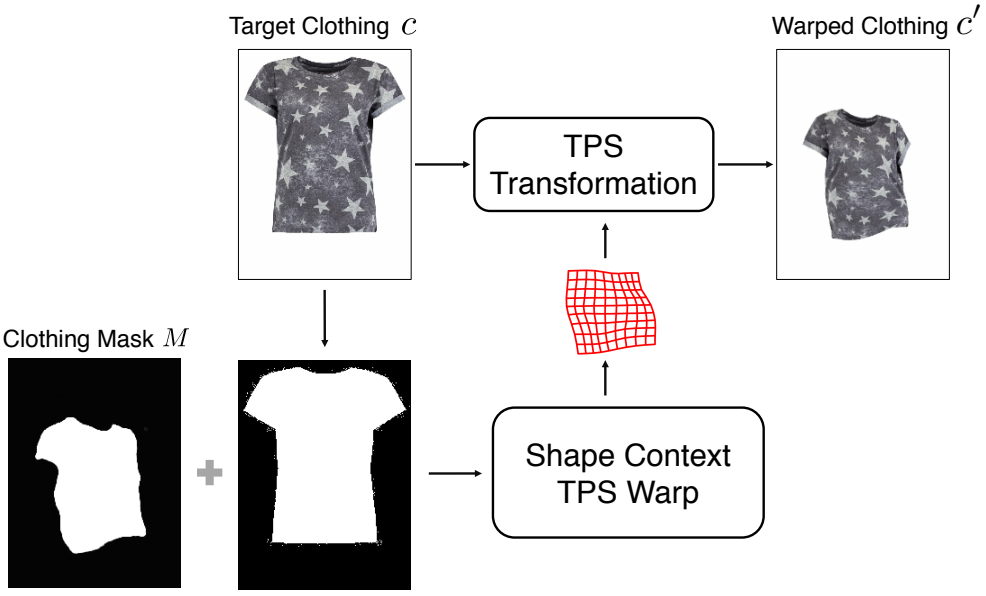


Figure 4.4: Warping a clothing image. Given the target clothing image and a clothing mask predicted in the first stage, we use shape context matching to estimate the TPS transformation and generate a warped clothing image.

image c to fill in the details in the generated region of the coarse sample. However, directly pasting the product image is not suitable as clothes deform conditioned on the person pose and body shape. Therefore, we warp the clothing item by estimating a thin plate spline (TPS) transformation with shape context matching [101], as illustrated in Figure 4.4. More specifically, we extract the foreground mask of c and compute shape context TPS warps [101] between this mask and the clothing mask M of the person, estimated with Eqn. 4.1. These computed TPS parameters are further applied to transform the target clothing image c into a warped version c'. As a result, the warped clothing image conforms to pose and body shape information of the person and fully preserves the details of the target item. The idea is similar

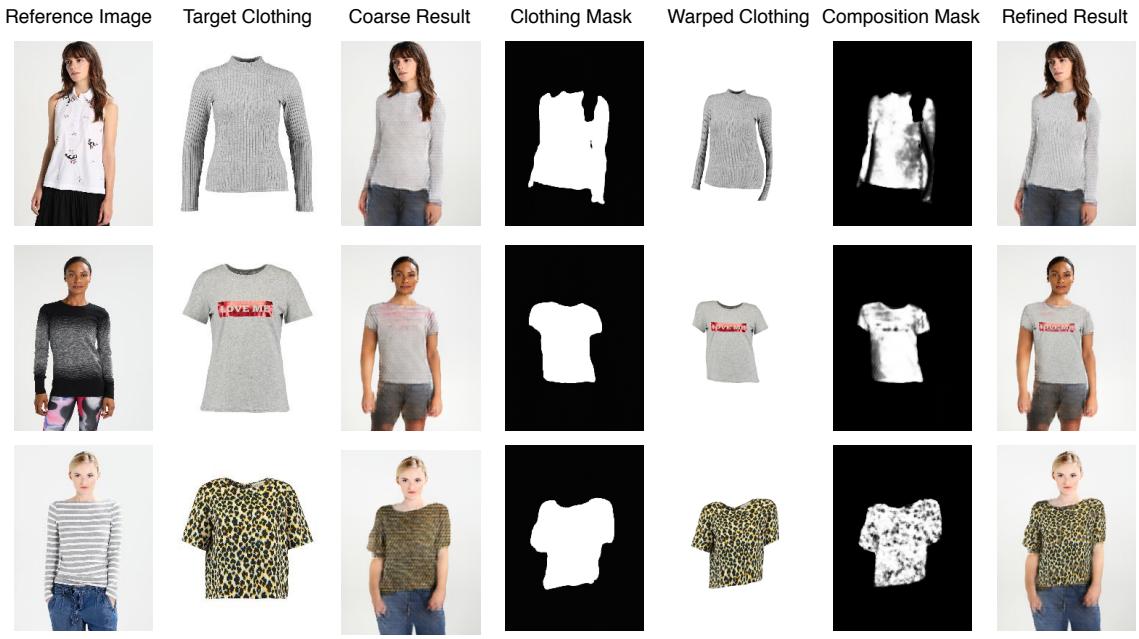


Figure 4.5: Output of different steps in our method. Coarse sythetic results generated by the encoder-decoder are further improved by learning a composition mask to account for details and deformations.

to recent 2D/3D texture warping methods for face synthesis [102, 103], where 2D facial keypoints and 3D pose estimation are utilized for warping. In contrast, we rely on the shape context-based warping due to the lack of accurate annotations for clothing items. Note that a potential alternative to estimating TPS with shape context matching is to learn TPS parameters through a Siamese network as in [104]. However, this is particularly challenging for non-rigid clothes, and we empirically found that directly using context shape matching offers better warping results for virtual try-on.

Learn to composite. The composition of the warped clothing item c' onto the coarse synthesized image I' is expected to combine c' seamlessly with the clothing

region and handle occlusion properly in cases where arms or hair are in front of the body. Therefore, we learn how to composite with a refinement network. As shown at the bottom of Figure 4.2, we first concatenate c' and the coarse output I' as the input of our refinement network G_R. The refinement network then generates a 1-channel composition mask $\alpha \in (0,1)^{m \times n}$, indicating how much information is utilized from each of the two sources, *i.e.*, the warped clothing item c' and the coarse image I'. The final virtual try-on output of VITON $\hat{I}$ is a composition of c' and I':

$$\hat{I} = \alpha \odot c' + (1 - \alpha) \odot I', \tag{4.2}$$

where $\odot$ represents element-wise matrix multiplication. To learn the optimal composition mask, we minimize the discrepancy between the generated result $\hat{I}$ and the reference image I with a similar perceptual loss L_{perc} as Eqn. 4.1:

$$L_{perc}(\hat{I}, I) = \sum_{i=3}^{5} \lambda_i ||\phi_i(\hat{I}) - \phi_i(I)||_1, \tag{4.3}$$

where ϕ denotes the visual perception network VGG19. Here we only use 'conv3_2', 'conv4_2', 'conv5_2' for calculating this loss. Since lower layers of a visual perception network care more about the detailed pixel-level information of an image instead of its content [105], small displacements between I and $\hat{I}$ (usually caused by imperfect warping) will lead to a large mismatch between the feature maps of lower layers ('conv1' and 'conv2'), which, however, is acceptable in a virtual try-on setting. Hence, by only using higher layers, we encourage the model to ignore the effects of imperfect warping, and hence it is able to select the warped target clothing image and preserve more details.

We further regularize the generated composition mask output by G_R with an L_1 norm and a total-variation (TV) norm. The full objective function for the refinement network then becomes:

$$L_{G_R} = L_{perc}(\hat{I}, I) - \lambda_{warp}||\alpha||_1 + \lambda_{TV}||\nabla\alpha||_1, \tag{4.4}$$

where λ_{warp} and λ_{TV} denote the weights for the L_1 norm and the TV norm, respectively. Minimizing the negative L_1 term encourages our model to utilize more information from the warped clothing image and render more details. The total-variation regularizer $||\nabla\alpha||_1$ penalizes the gradients of the generated composition mask α to make it spatially smooth, so that the transition from the warped region to the coarse result looks more natural.

Figure 4.5 visualizes the results generated at different steps from our method. Given the target clothing item and the representation of a person, the encoder-decoder produces a coarse result with pose, body shape and face of the person preserved, while details like graphics and textures on the target clothing item are missing. Based on the clothing mask, our refinement stage warps the target clothing image and predicts a composition mask to determine which regions should be replaced in the coarse synthesized image. Consequentially, important details (material in the 1st example, text in 2nd example, and patterns in the 3rd example) "*copied*" from the target clothing image are "*pasted*" to the corresponding clothing region of the person.

4.4 Experiments

4.4.1 Dataset

The dataset used in [93] is a good choice for conducting experiments for virtual try-on, but it is not publicly available. We therefore collected our own dataset. We first crawled around 19,000 frontal-view woman and top[1] clothing image pairs and then removed noisy images with no parsing results, yielding 16,253 pairs. The remaining images are further split into a training set and a testing set with 14,221 and 2,032 pairs respectively. Note that during testing, the person should wear a different clothing item than the target one as in real-world scenarios, so we randomly shuffled the clothing product images in these 2,032 test pairs for evaluation.

4.4.2 Implementation Details

Training setup. Following recent work using encoder-decoder structures [84, 93], we use the Adam [106] optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$, and a fixed learning rate of 0.0002. We train the encoder-decoder generator for 15K steps and the refinement network for 6K steps both with a batch size of 16. The resolution of the synthetic samples is 256×192.

Encoder-decoder generator. Our network for the coarse stage contains 6 convolutional layers for encoding and decoding, respectively. All encoding layers consist

[1]Note that we focus on tops since they are representative in attire with diverse visual graphics and significant deformations. Our method is general and can also be trained for pants, skirts, outerwears, *etc.*

Figure 4.6: Qualitative comparisons of different methods. Our method effectively renders the target clothing on to a person.

of 4×4 spatial filters with a stride of 2, and their numbers of filters are 64, 128, 256, 512, 512, 512, respectively. For decoding, similar 4×4 spatial filters are adopted with a stride of 1/2 for all layers, whose number of channels are 512, 512, 256, 128, 64, 4. The choice of activation functions and batch normalizations are the same as in [75]. Skip connections [96] are added between encoder and decoder to improve the performance. λ_i in Eqn. 4.1 is chosen to scale the loss of each term properly [87].

Refinement network. The network is a four-layer fully convolutional model. Each of the first three layers has $3 \times 3 \times 64$ filters followed by Leaky ReLUs and the last

layer outputs the composition mask with 1×1 spatial filters followed by a sigmoid activation function to scale the output to $(0, 1)$. λ_i in Eqn. 4.4 is the same as in Eqn. 4.1, $\lambda_{warp} = 0.1$ and $\lambda_{TV} = 5e - 6$.

Runtime. The runtime of each component in VITON: Human Parsing (159ms), Pose estimation (220ms), Encoder-Decoder (27ms), TPS (180ms), Refinement (20ms). Results other than TPS are obtained on a K40 GPU. We expect further speed up of TPS when implemented in GPU.

4.4.3 Compared Approaches

To validate the effectiveness of our framework, we compare with the following alternative methods.

GANs with Person Representation (PRGAN) [78,89]. Existing methods that leverage GANs conditioned on either poses [78] or body shape information [89] are not directly comparable since they are not designed for the virtual try-on task. To achieve fair comparisons, we enrich the input of [78,89] to be the same as our model (a 22-channel representation, p + target clothing image c) and adopt their GAN structure to synthesize the reference image.

Conditional Analogy GAN (CAGAN) [93]. CAGAN formulates the virtual try-on task as an image analogy problem - it treats the original item and the target clothing item together as a condition when training a Cycle-GAN [79]. However, at test time, it also requires the product image of the original clothing in the reference image, which makes it infeasible in practical scenarios. But we compare with this

approach for completeness. Note that for fairness, we modify their encoder-decoder generator to have the same structure as ours, so that it can also generate 256×192 images. Other implementation details are the same as in [93].

Cascaded Refinement Network (CRN) [87]. CRN leverages a cascade of refinement modules, and each module takes the output from its previous module and a downsampled version of the input to generate a high-resolution synthesized image. Without adversarial training, CRN regresses to a target image using a CNN network. To compare with CRN, we feed the same input of our generator to CRN and output a 256×192 synthesized image.

Encoder-decoder generator. We only use the network of our first stage to generate the target virtual try-on effect, without the TPS warping and the refinement network.

Non-parametric warped synthesis. Without using the coarse output of our encoder-decoder generator, we estimate the TPS transformation using shape context matching and paste the warped garment on the reference image. A similar baseline is also presented in [89].

The first three state-of-the-art approaches are directly compared with our encoder-decoder generator without explicitly modeling deformations with warping, while the last Non-parametric warped synthesis method is adopted to demonstrate the importance of learning a composition based on the coarse results.

Figure 4.7: Effect of removing pose and body shape from the person representation. For each method, we show its coarse result and predicted clothing mask output by the corresponding encoder-decoder generator.

4.4.4 Qualitative Results

Figure 4.6 presents a visual comparison of different methods. CRN and encoder-decoder create blurry and coarse results without knowing where and how to render the details of target clothing items. Methods with adversarial training produce shaper edges, but also cause undesirable artifacts. Our Non-parametric baseline directly pastes the warped target image to the person regardless of the inconsistencies between the original and target clothing items, which results in unnatural images. In contrast to these methods, VITON accurately and seamlessly generates detailed virtual try-on results, confirming the effectiveness of our framework.

However, there are some artifacts around the neckline in the last row, which

results from the fact that our model cannot determine which regions near the neck should be visible (*e.g.*, the neck tag should be hided in the final result, see supplementary material for more discussions). In addition, pants, without providing any product images of them, are also generated by our model. This indicates that our model implicitly learns the co-occurrence between different fashion items. VITON is also able to keep the original pants if the pants regions are handled in the similar way as face and hair (*i.e.*, extract pants regions and take them as the input to the encoder). More results and analysis are present in the supplementary material.

Person representation analysis. To investigate the effectiveness of pose and body shape in the person representation, we remove them from the representation individually and compare with our full representation. Sampled coarse results are illustrated in Figure 4.7. We can see that for a person with a complicated pose, using body shape information alone is not sufficient to handle occlusion and pose ambiguity. Body shape information is also critical to adjust the target item to the right size. This confirms the proposed clothing-agnostic representation is indeed more comprehensive and effective than prior work.

Failure cases. Figure 4.8 demonstrates two failure cases of our method due to rarely-seen poses (example on the left) or a huge mismatch in the current and target clothing shapes (right arm in the right example).

In the wild results. In addition to experimenting with constrained images, we also utilize in the wild images from the COCO dataset [107], by cropping human body regions and running our method on them. Sample results are shown in Figure 4.9, which suggests our method has potentials in applications like generating people

Figure 4.8: Failure cases of our method.

Figure 4.9: In the wild results. Our method is applied to images on COCO.

in clothing [74].

Method	IS	Human
PRGAN [78, 89]	2.688 ± 0.098	27.3%
CAGAN [93]	2.981 ± 0.087	21.8%
CRN [87]	2.449 ± 0.070	69.1%
Encoder-Decoder	2.455 ± 0.110	58.4%
Non-parametric	3.373 ± 0.142	46.4%
VITON (Ours)	2.514 ± 0.130	77.2%
Real Data	3.312 ± 0.098	-

Table 4.1: Quantitative evaluation on dataset.

4.4.5 Quantitative Results

We also compare VITON with alternative methods quantitatively based on Inception Score [108] and a user study.

Inception Score. Inception Score (IS) [108] is usually used to quantitatively evaluate the synthesis quality of image generation models [77, 78, 109]. Models producing visually diverse and semantically meaningful images will have higher Inception Scores, and this metric correlates well with human evaluations on image datasets like CIFAR10.

Perceptual user study. Although Inception Score can be used as an indicator of the image synthesis quality, it cannot reflect whether the details of the target clothing are naturally transferred or the pose and body of the clothed person are

preserved in the synthesized image. Thus, simialr to [87, 110], we conducted a user study on the Amazon Mechanical Turk (AMT) platform. On each trial, a worker is given a person image, a target clothing image and two virtual try-on results generated by two different methods (both in 256×192). The worker is then asked to choose the one that is more realistic and accurate in a virtual try-on situation. Each AMT job contains 5 such trials with a time limit of 200 seconds. The percentage of trials in which one method is rated better than other methods is adopted as the Human evaluation metric following [87] (chance is 50%).

Quantitative comparisons are summarized in Table 4.1. Note that the human score evaluates whether the virtual try-on results, synthetic images with a person wearing the target item, are realistic. However, we don't have such ground-truth images - the same person in the same pose wearing the target item (IS measures the characteristics of a set, so we use all reference images in the test set to estimate the IS of real data).

According to this table, we make the following observations: (a) Automatic measures like Inception Score are not suitable for evaluating tasks like virtual try-on. The reasons are two-fold. First, these measures tend to reward sharper image content generated by adversarial training or direct image pasting, since they have higher activation values of neurons in Inception model than those of smooth images. This even leads to a higher IS of the Non-parametric baseline over real images. Moreover, they are not aware of the task and cannot measure the desired properties of a virtual try-on system. For example, CRN has the lowest IS, but ranked the 2nd place in the user study. Similar phenomena are also observed in [87, 98]; (b)

Person representation guided methods (PRGAN, CRN, Encoder-Decoder, VITON) are preferred by humans. CAGAN and Non-parametric directly take the original person image as inputs, so they cannot deal with cases when there are inconsistencies between the original and target clothing item, *e.g.*, rendering a short-sleeve T-shirt on a person wearing a long-sleeve shirt; (c) By compositing the coarse result with a warped clothing image, VITON performs better than each individual component. VITON also obtains a higher human evaluation score than state-of-the-art generative models and outputs more photo-realistic virtual try-on effects.

To better understand the noise of the study, we follow [78,87] to perform time-limited (0.25s) real or fake test on AMT, which shows 17.18% generated images are rated as real, and 11.46% real images are rated as generated.

4.5 Conclusion

We presented a virtual try-on network (VITON), which is able to transfer a clothing item in a product image to a person relying only on RGB images. A coarse sample is first generated with a multi-task encoder-decoder conditioned on a detailed clothing-agnostic person representation. The coarse results are further enhanced with a refinement network that learns the optimal composition. We conducted experiments on a newly collected dataset, and promising results are achieved both quantitatively and qualitatively. This indicates that our 2D image-based synthesis pipeline can be used as an alternative to expensive 3D based methods.

Chapter 5: Face-swap with a Perceptually-aware Discriminator

5.1 Introduction and related works

Generative networks (*e.g.*, encoder-decoder neural networks) [75,82] have been proven very effective in synthesizing novel images with desired properties given various input conditions like class labels [77], text [73], attributes [86], poses [78], *etc.* In this part of the dissertation, we investigate how to utilize generative adversarial networks for fast face-swap - changing an input face to a target identity. Two face-swap examples are illustrated in Figure 5.1.

This problem is original proposed and addressed in [111]. Given an input face, they select a target face forming a large face library that has the similar resolution, image blur, lighting, and seam signature to the input face. Then the color and lighting are further adjusted for the target face and used to replace the input face. This technique can be used to applications like face de-identification, composite group photographs, and other creative scenarios. However, one cannot control the target identity and the facial expression present in the synthesized image.

To tackle these two limitations, Korshunova *et al.* [8] borrow the idea used in artistic style transfer [97, 105, 112] and regard each target identity as a new style and train a convolutional neural network for swapping faces. More specifically, for

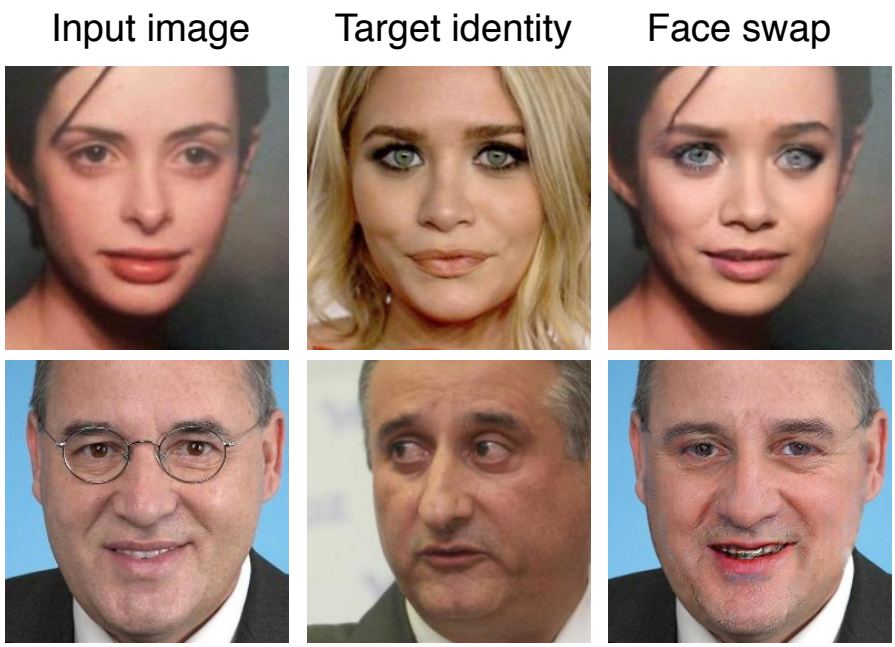


Figure 5.1: Our goal is to replace the face in the input image to a target identification while keeping its original facial expressions. The first column is the input image, our alogrithm swap the identity in the second column to the input image.

each individual target identity (style), they train a feed-forward neural network to enforce the output have similar identity as the target, they further propose a lighting loss to match the lighting conditions between faces. This method still have two drawbacks: firstly, it needs to train a separate neural network for each target identity, which limits their scalability; secondly, without using identity reserving loss and adversarial training, the results sometimes look unrealistic and lose the target identity information.

In this chapter, we propose a GAN based method for generating realistic and identity guided face-swapping results. The framework of our method is shown in

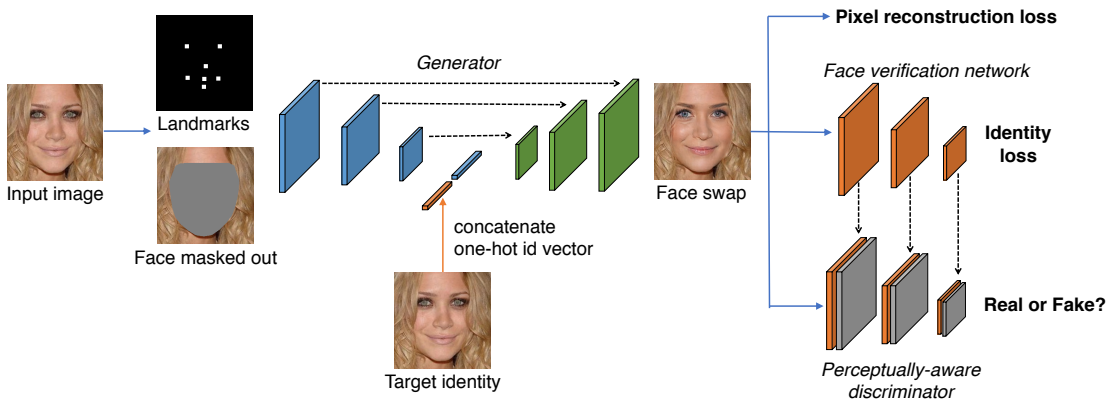


Figure 5.2: Training framework of our proposed face-swap method. For an input face, its key facial landmarks and face mask are extracted as input to an encoder-decoder generator. The target identity is encoded as a one-hot vector and concatenated to the bottleneck of the generator to inject guidance of identity in the generation process. An L_1 pixel loss encourages good reconstruction of the input image, and an identity loss built on a pre-trained face verification network makes the generated face perceptually similar to the input image. The representations of intermediate layers of the face verification network are then concatenated to the discriminator to ensure the results look realistic by incorporating some face-specific features.

Figure 5.2. During training, given an input face, we input its landmarks and face mask (extracted by [113]) to our face generator. We then embed the target identity by a one-hot vector concatenated to the bottleneck of the generator to guide the generation process. Three different losses are utilized to generate satisfactory results: (1) an L_1 pixel reconstructs the input image; (2) a perceptual loss that uses a

network trained for large-scale face classification as a feature extractor, and penalizes the feature distances of different layers for the input and generated faces; (3) an adversarial loss that is modeled by a discriminator, in which intermediate layer features are concatenated to corresponding feature maps of the discriminator to incorporate higher level perceptual information into the network. At test time, the input and target identity are different, and our network will synthesize the target identity on to the input face.

The proposed method is highly related to recent advances developed to synthesize face images using GANs. Recently, Huang *et al.* [9] propose a dual-path generative adversarial network that jointly synthesizes a whole frontal face and different local patches on the face. The identity of the generated face is then preserved by a face verification network. Zhao *et al.* [114] leverage two discriminators, one classifies the identity and the other distinguishes fake faces from real ones. They use the generated faces of different view angles to achieve state-of-the-art performance on face verification tasks. Tran *et al.* [115] disentangle the pose and identity of a face in a GAN to rotate faces. However, these methods are still not able to control the target identity.

It is interesting to note that our work share the similar problem settings as VITON [116] described in the previous chapter. In VITON, we try to swap a target clothing item on to a person while keep the person's pose unchanged, which is achieved by giving the target clothing item and pose keypoints as priors to the network. For face-swap task, we also inject facial keypoints and identity information to the network and swap faces rather than clothing. The main difference is that

people's vision system is much more sensitive to faces than clothing, thus a more carefully designed discriminator is needed for generating photorealistic results.

The contributions of our approach are two-folds: First, by utilizing a GAN framework, our method can replace a face with desired identity by one forward pass of the generator, while traditional approaches cannot control the swapped identity or need to train an individual network for each target identity. Moreover, in contrast to existing work that directly feeds an image to a discriminator to evaluate if an image looks realistic, we propose a perceptually-aware discriminator where mid-level and high-level perceptual information is considered by including features of different layers of the pre-trained face verification network in the discriminator.

5.2 GAN with Perceptually-aware Discriminator

There are three main parts in our proposed model: a face generator, a pre-trained face verification network, and a perceptually-aware discriminator. We will describe them in details.

5.2.1 Face Generator

Inspired by recent advances of GAN and its variations [73, 75, 78, 116], we also employ an encoder-decoder [96] generator (denoted as G) for training the synthesizing network. To be specific, given an aligned face image x, we first extract its 68 keypoints using Dlib library [117] based on [113]. Similar to [116], there should be no identity information in the input of the generator since during testing we want

to swap faces. To this end, we use the extracted keypoints to compute two crucial identity-agnostic representations of x:

Masked out face. When synthesizing a face we want to keep the information like skin color, hair style, etc., the same as the input face, so we use the convex hull of the face landmarks to erase the face regions and retain the other parts including forehead, hairs. Given this information, the network can learn to inpaint the missing part naturally.

Face landmarks. Since we want to preserve the face expression and view point of the input face after swapping, we also include 7 key landmarks of the input face in the input to the generator - two are the centers of two eyes, one for the nose, and 4 (left, right, top, bottom) points at the mouth contour. We do not use all 68 landmarks because using all landmarks will affect synthesizing the desired identity. For example, there are 6 landmarks around an eye, which contain the information about the shape of eyes. If this information is used in the generator, the generator will enforce the generated eye have the same shape as the input person and hurt the synthesizing performance. In sum, these 7 landmarks give guidance to the locations of eyes/nose/mouth position and the view point of the face, so the generator can preserve the input facial expression.

For effectively controlling the generated identity, we follow [115] and encode the target identity as a one-hot identity vector then concatenate it to the latent feature space of the generator. The length of this vector equals to the number of identities used during training (denoted as N_i). As a result, the network learns to generate different identities according to the given target identity vector. We

denote the generated image as $G(x, v)$, where v is the target identity vector. During training, the identity vector v corresponds to the input identity in x, *i.e.* $v = v_x$. The generator minimizes the following pixel L_1 loss:

$$L_{pixel} = ||G(x, v_x) - x||_1, \tag{5.1}$$

The generator is similar to the one used in the previous chapter [116] except that the input size is changed to 256×256 and there a concatenated vector in the bottleneck.

5.2.2 Face Verification Network

The most crucial property of a face-swap system is that the generated face should present the desired identity, so only focusing on reconstructing the face in pixel domain by Eqn. 5.1 is not enough. Thus, we need to minimize the distance of output face and target face in an embedding space which accurately measures the identity information. Inspired by perceptual loss [97] that uses a pre-trained network and minimizes the distance of intermediate layer features of the input and output image to maintain the content of the output, [9] utilizes the similar technique for preserving identity when synthesizing frontal faces. Similarly, we use a ResNet50 face verification network trained on a large face dataset [118] as the perceptual network, and then enforce the identity to be preserved by minimizing:

$$L_{id} = \sum_{i=3}^{5} \lambda_i ||\phi_i(G(x, v_x)) - \phi_i(x)||_1, \tag{5.2}$$

where $\phi_i(y)$ the feature representation of face x of the i-th layer in the face verification network ϕ, we utilize 'conv3', 'conv4', 'conv5' of the ResNet model. The balancing weights λ_i controls the contribution of the i-th layer to the total loss. By minimizing Eqn. 5.2, the encoder-decoder learns to generate the target identity conditioned on v. These features of the synthesized image are further utilized to incorporate rich information in the discriminator.

5.2.3 Perceptually-aware Discriminator

A discriminator is used in GANs to help generate realistic images. The discriminator network D takes a real face image x as input and tries to classify it as real. While given a synthesized image $\hat{x} = G(x, v)$ from the generator, D aims to classify it as a fake image. This is achieve by minimizing the adversarial loss:

$$L_{adv} = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{\hat{x} \sim p_G(\hat{x})}[log(1 - D(\hat{x}))], \tag{5.3}$$

However, it is worth noting that the traditional discriminator simply takes an image as input without incorporating any prior knowledge about faces. This hurts the performance of face-swap, where face/identity specific features are needed to enforce realism and preserve the desired identity. Hence, we leverage the high-level perceptual information in the pre-trained face classification network ϕ by concatenating representations of ϕ's layers to the corresponding layers (layers with same feature map resolutions) of a traditional discriminator. Orange and gray blocks in Figure 5.2 illustrate this design. Now, our adversarial loss becomes:

$$L_{adv} = \mathbb{E}_{x \sim p_{data}(x)}[logD(x, \phi(x))] + \mathbb{E}_{\hat{x} \sim p_G(\hat{x})}[log(1 - D(\hat{x}, \phi(\hat{x}))], \tag{5.4}$$

By minimizing this loss D takes not only the pixel-level value of an image but also multi-scale and multi-level face-specific features to better distinguish fake faces from real ones. This pyramid design is very common in object detection [119] or semantic segmentation [120] but has not well explored in a GAN setting. Our experiments show that this design helps create more realistic and identity-preserving faces. Note that the parameters of ϕ are fixed in our approach.

5.2.4 Loss Function

Finally, our face-swap approach optimizes the following objective function:

$$\begin{cases} L_G = -L_{adv} + \lambda_{id} L_{id} + \lambda_{pixel} L_{pixel} \\ L_D = L_{adv} \end{cases}, \tag{5.5}$$

which is optimized by alternatively minimizing L_G and L_D. Note that ϕ, present both in the generator and discriminator loss, injects face and identity related features into our framework.

5.3 Experimental Evaluation

5.3.1 Experimental Settings

Dataset. We train our face-swap network on VGGFace2 [118] dataset. VGGFace2 contains 3.31M images of 9,131 subjects (8,631 for training, 500 for testing).

Since images of high resolution work better to train a GAN, we select the subjects containing more than 100 faces that are larger than 256×256. This results 403 subjects and 51,811 faces larger than 256×256, in which we use 39,004 images of 303 subjects (*i.e.*, $N_{id} = 303$) for training, and 12,807 images of 100 subjects for testing. At test time, the target identity one-hot vector corresponds to one random chosen identity in the 303 training identities.

Since there are few work on this task, we compare our method with our baseline - a traditional GAN without our proposed perceptually-aware discriminator. We use face classification accuracy for evaluating the identity preservation performance.

Implementation details. The training setup is almost identical to [116], we train our network with Adam [106] optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate is set to 0.0002. Training takes 60K steps to converge. The input faces are aligned using [113] and resized to 256×256 before inputing into the network, and the output face size is also 256×256.

The generator network contains 7 convolutional layers for encoding and decoding, respectively. All encoding layers consist of 4×4 spatial filters with a stride of 2, and their numbers of filters are 64, 128, 256, 512, 512, 512, 512, respectively. For decoding, similar 4×4 spatial filters are used for deconvolution, whose number of channels are 512, 512, 512, 256, 128, 64, 3. Leaky ReLu and batch normalizations are the same as in [75, 116]. Skip connections [96] are added between encoder and decoder to improve the performance.

The face verification network is a ResNet50 network [121], whose 'conv3', 'conv4' and 'conv5' layers produce feature maps of size $32 \times 32 \times 512$, $16 \times 16 \times$

1024, and $8 \times 8 \times 2048$ with a 256×256 input image. We use a publicly available implementation [1] trained on VGGFace2 dataset.

The discriminator also have 7 convolutional layers with a similar structure as the encoder of the generator, whose numbers of output channels are 64, 128, 256, 512, 512, 512, 1. The output of its 3th, 4th, 5th layers are concatenated with 'conv3' (512 channels), 'conv4' (1024 channels), 'conv5' (2048 channels) layers of ResNet50 as the input to the 4th, 5th and 6th layers respectively. As a result, the input channels of the 4th, 5th and 6th layers in the discriminator layers are 1024 (512+512), 1536 (512+1024), and 2564 (512+2048).

5.3.2 Qualitative Results

Perceptually-aware discriminator improves the quality. A vanilla discriminator used in traditional GANs merely takes the pixel values of an image and tries to distinguish fake ones from real ones. This leads to loss of face-specific information and sometimes yields unrealistic faces and unsatisfactory artifacts. In Figure 5.3, we can find that without perceptual information in the discriminator, a vanilla GAN produces unrealistic faces. For example, as shown in the second column of Figure 5.3, some asymmetric eye artifacts make the results unappealing. In the first two examples, left eye is larger than the right eye; and in the last example, eyes have different colors. In contrast, our method eliminates these artifacts by telling the discriminator what a natural and real face should look like.

Figure 5.4 represents more visual results of our approach, in which we can find

[1] `https://github.com/rcmalli/keras-vggface`

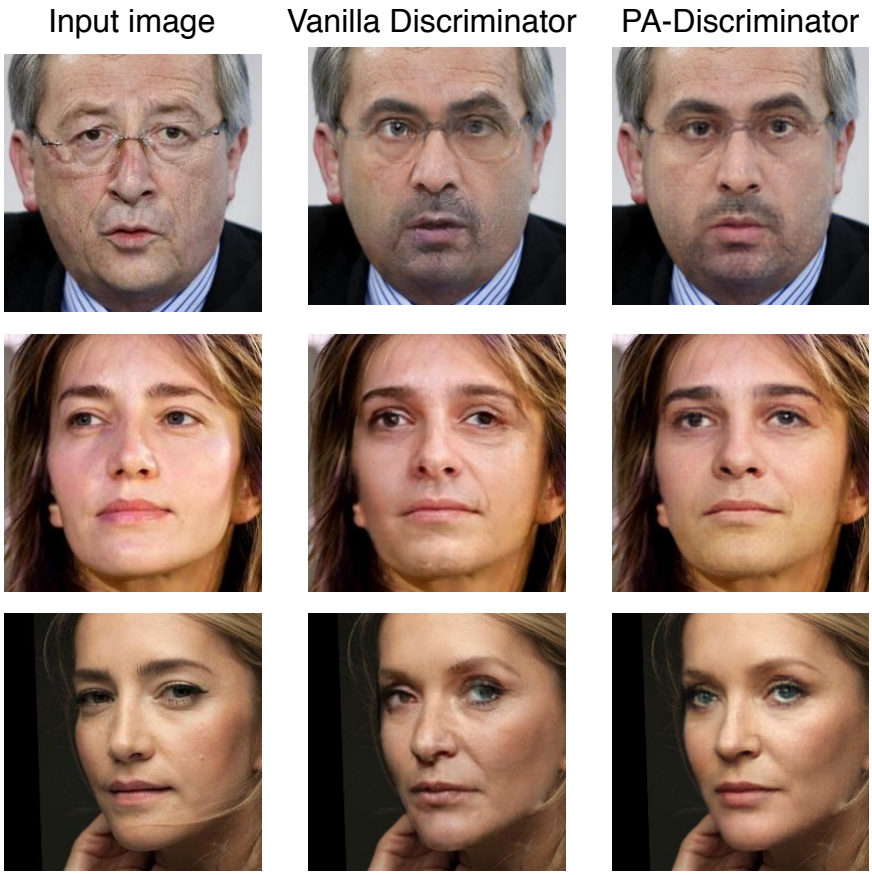


Figure 5.3: Visual comparison of our GAN with and without perceptually-aware discriminator. A traditional discriminator losses face-specific information and creates artifacts like asymmetric eyes. By incorporating rich features learned from a face verification network into the discriminator, our proposed perceptually-aware (PA-discriminator) generator effectively avoids these artifacts.

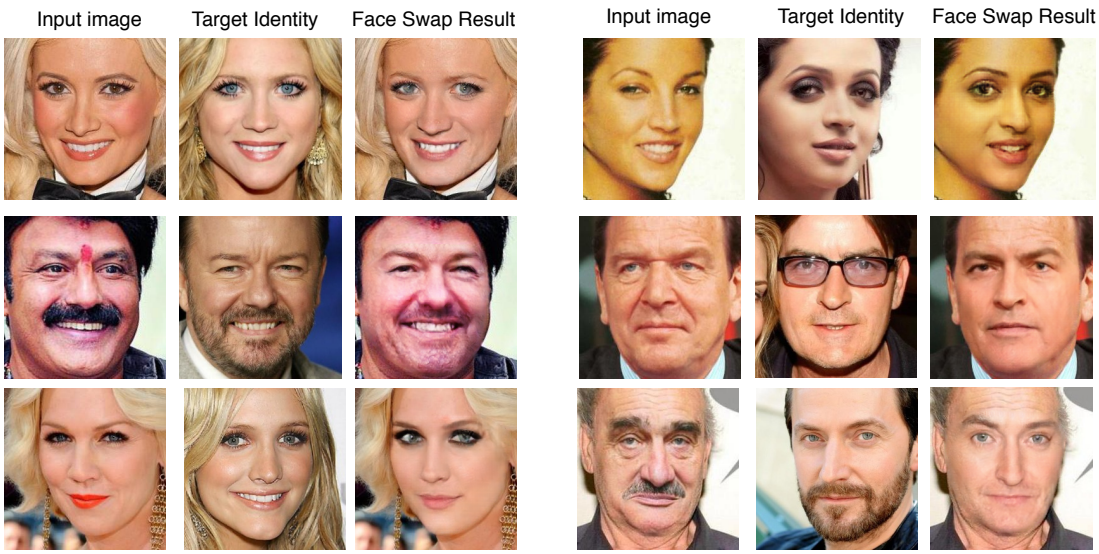


Figure 5.4: Examples of our face-swap results. Note that target identity is illustrated by an example image of that identity for the sake of visualization, and our system did not take these images as input.

our method can successfully generates realistic and vivid face-swap results.

5.3.3 Identity Preserving Performance

As mentioned, one critical property of a face-swap method is to preserve the target identity information. We evaluate the identity preservation performance using a face classification network trained on VGGFace2 dataset. The results are shown in Table 5.1. From this table, we can see that by using our perceptually-aware discriminator, our model can generate faces that are more recognizable as the target identity. Also, we should notice that this is a pretty challenging task, since the face classification network is trained to classify 8,631 different identity (chance is only 0.0116%). Although our method still has a gap to perfect, we believe our work is

Table 5.1: Top-k classification accuracy of swapped faces. Our method achieves higher recognition accuracy than its baseline.

	top-1 acc.	top-3 acc.	top-5 acc.
Regular discriminator	0.347	0.469	0.524
Perceptually-aware discriminator (Ours)	**0.371**	**0.486**	**0.541**

an important step towards perfect face-swap algorithm.

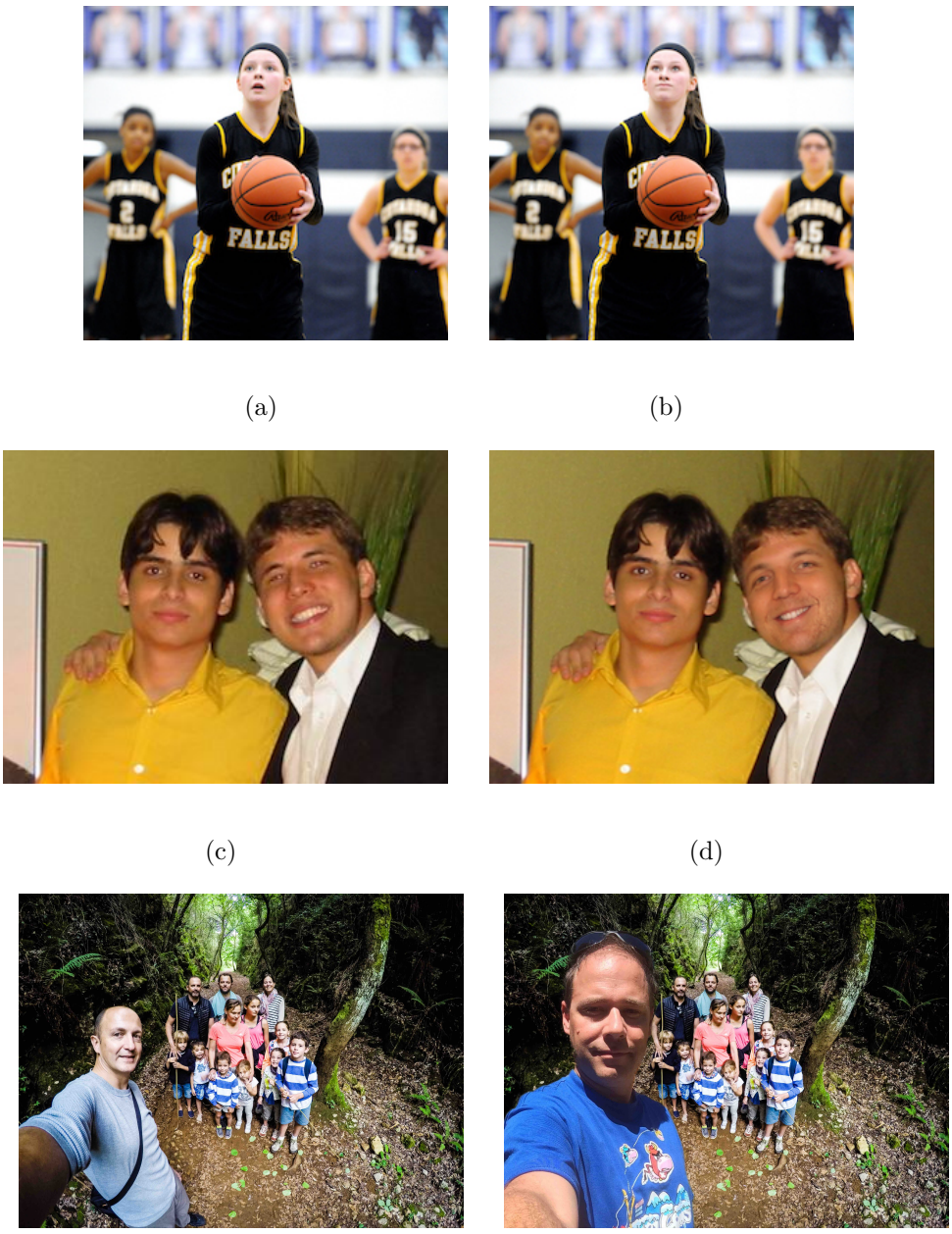
5.4 Conclusion

In this part of the dissertation, we describe how to utilize a generative adversarial network to create realistic and identity preserving face-swap results. We extract an identity-independent representation of a face as the input to the generator and embed the target identity information in the bottleneck latent space of the generator. To ensure the swapped face presents the desired identity and look realistic, we further added a face verification loss and a perceptually-aware discriminator. Qualitative and quantitative results show that the proposed method can swap faces while controlling the target identity.

Chapter 6: Learning high-level and low-level features for tampered person detection

6.1 Introduction

People post photos every day on popular social websites such as Facebook, Instagram and Twitter. A considerable number of these photos are authentic as they are generated from people's real life and shared as a part of their social experience. However, maliciously or not, more and more tampered images, especially ones involving face regions, are emerging on the Internet. Image splicing, which is the most common tampering manipulation, is the process of cutting one part of a source image, such as the face or body regions, and inserting it in the target image. To make the tampered result more realistic, adjustments on the shape, boundary, illumination and scaling are necessary, which make tamper detection challenging. Given advances in face detection and recognition techniques, anyone is able to swap faces with low cost using mobile applications [122] or open-source software [7]. Besides face-swap, spliced person (portrait) images are also very common among manipulated images using Photoshop or other photo editing tools. Some tampered image examples generated from commercial software are shown in Figure 6.1. Even after



(a) (b)

(c) (d)

(e) (f)

Figure 6.1: Examples of tampered faces. (a) original image. (b) Tampered image. The face in the middle has been tampered. (c) Original image. (d) Tampered image. The face on the right has been tampered. (e) Original image. (f) Tampered image. The person on the left is copied and pasted from anthoer image.

close inspection, people mistake the tampered images as original. The consequence would be even more serious if manipulated images are used for political or commercial purposes.

Even though image tampering detection has been an active research area during the last decade, limitations still exist for current approaches since they focus on a particular source of tampering evidence [123–126]. For example, local noise analysis fails to deal with tampered images constructed using careful post processing and Color Filter Array (CFA) models cannot deal with resized images. To avoid focusing on specific tampering evidence and achieve robust tampering detection, we proposed to learn both high-level and low-level tampering artifacts. According to different manipulation categories, we focuses on two tasks: (1) *Tampered face detection* and (2) *spliced portrait detection*.

6.1.1 Tampered face detection

For tampered face detection, we propose a two-stream network architecture to capture both tampering artifact evidence and local noise residual evidence as shown in Figure 6.2. This is inspired by recent research on CNNs showing the potential to learn tampering evidence [127–129] and rich models [130–133], which are models of the noise components that produce informative features for steganalysis and we call the features produced by these models "steganalysis features" in the rest of this dissertation, showing good performance on tampering detection. One of our streams is a CNN based face classification stream and the other one is a

steganalysis feature based triplet stream. The face classification stream, based on GoogLeNet [134], is trained on tampered and authentic images and serves as a tampered face classifier. The patch triplet stream, based on patch level steganalysis features [130, 133], captures low-level camera characteristics like CFA pattern and local noise residuals. Instead of utilizing steganalysis features directly, we train a triplet network after extracting steganalysis features to allow the model to refine steganalysis features. Combining the two streams reveals both evidence of high-level tampering artifacts and low-level noise residual features, and yields very good performance for face tamper detection.

To train and evaluate our approach, we created a new face tampering dataset. The new dataset overcomes the drawback of existing datasets [126, 135–137] that are either small or in which the tampering is easily distinguishable even through visual inspection. We chose two face swapping apps to create two parallel sets of tampered images where the same target face is swapped with the same source face, but using different swapping algorithms. Only tampered images of good quality were retained. There are 1005 tampered images for each tampering technique (2010 tampered images in total) and 1400 authentic images for each subset.

6.1.2 Tampered portrait detection

However, for spliced portrait detection, the aforementioned two-stream network cannot be directly applied. The reason behind it is that a tampered face usually occupies a small region of an image, and a large portion of the image are au-

thentic, thus training a triplet network on patches can be used to tell the probability of a patch comes from the image by comparing its representation in the embedding space with other patches. However, spliced portrait regions usually have similar area as the background, which makes the patch triplet stream less effective.

To address this limitation, we propose to detect tampered portraits by modeling different traces in an image. The framework is shown in Figure 6.3. Three models are fused to detect if a portrait has been manipulated or not: (i) PortraintNet: A binary classifier fine-tuned on ImageNet pre-trained GoogLeNet, which is similar as the face classification stream in the two-stream tampered face detection framework; (ii) SegNet: A U-Net predicts tampered masks and boundaries, followed by a LeNet (not shown in the figure) to classifier if the predicted masks and boundaries indicating the image has been tampered with or not; (iii) EdgeNet: A U-Net predicts the edge mask of a portrait, and the extracted edges goes through a GoogLeNet for tampering classification. Similar to tampered faces, there are actually few dataset that focuses on spliced portrait detection, so we created a dataset using foreground of a portrait matting dataset [3] and Adobe Photoshop for testing purposes.

Our contribution is two-fold. First, by explicitly modeling different possible aspects of tampering, our method learns both tampering artifacts and local noise residual features. In addition, we create new challenging datasets specific to face and person region tampering detection and conduct extensive experiments on these datasets to demonstrate the effectiveness of our proposed methods.

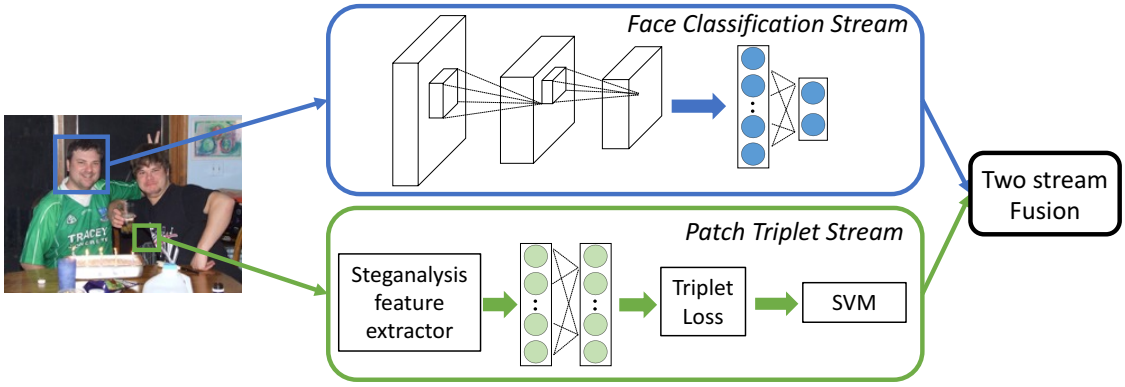


Figure 6.2: Illustration of our two-stream network. The face classification stream models visual appearance by classifying a face is tampered or not. The patch triplet stream is trained on steganalysis features to ensure patches from the same image are close in the embedding space, and an SVM trained on the learned features classifies each patch. Finally, the scores of two streams are fused to recognize a tampered face.

6.2 Related Work

There is growing research activity on tampering detection and localization. Prior methods can be classified according to the image features that they target, such as local noise estimation [123], double JPEG localization [125], CFA pattern analysis [124], illumination model [126] and steganalysis feature classification [138]. Recently, some methods based on Convolutional Neural Networks (CNN) [127–129] have achieved very good results.

The premise behind local noise estimation based techniques is that the difference between global noise characteristics and local noise characteristics reveals

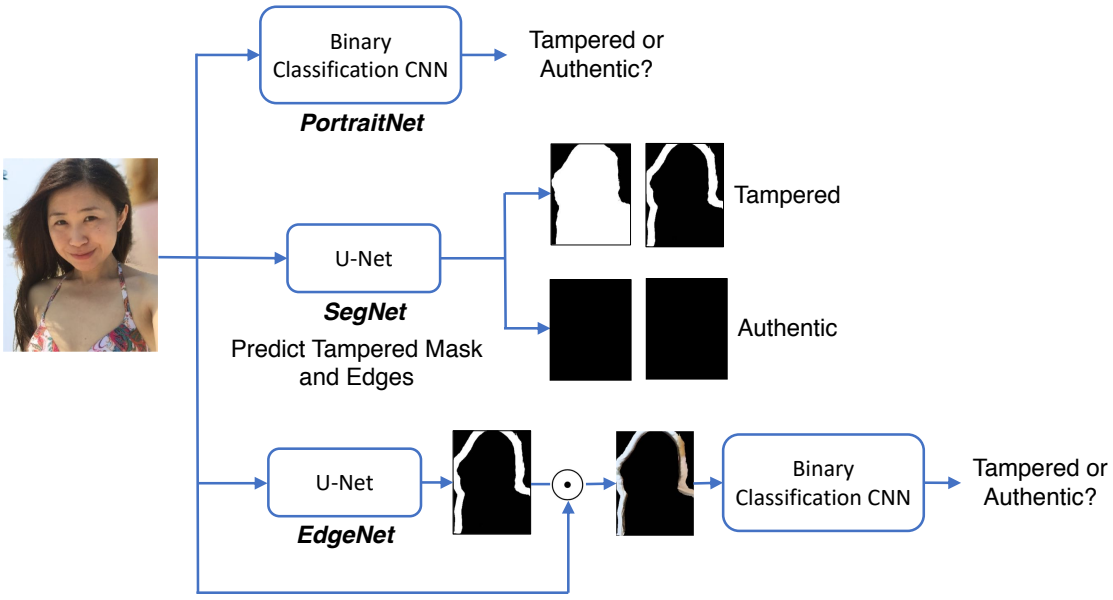


Figure 6.3: Illustration of our portrait detection approach. It contains three main modules. PortraitNet is a binary CNN captures visual appearance by classifying a portrait image is tampered or not. SegNet aims to predict tampered masks and edges for manipulated and predict blank images for authentic ones. The EdgeNet extract edges of the portrait and forces the network to only look at edges for making decision. Finally, the scores of each models are fused to obtain a detection score of a portrait.

the hidden tampered regions. For example, Lyu *et al.* [139] cast noise statistic estimation as a closed-form optimization problem. By exploiting the property of kurtosis of natural images in band-pass domains and the relationship between noise characteristics and kurtosis, they reveal the inconsistency between global noise and local noise. However, the assumption that local noise is inconsistent with global noise fails if post processing techniques like filtering or image blending are applied

after splicing. In contrast, our method learns local noise residuals that provide more reliable features for detection.

Double JPEG localization techniques can be classified into aligned double JPEG compression and non-aligned double JPEG compression, depending on whether or not the quantization factors align well after two applications of JPEG compression to the same image. The assumption is that the background regions undergo double JPEG compression while the tampered regions do not. For example, Bianchi *et al.* [125] presented a probability model that estimates DCT coefficients together with quantization factors. The advantage of this method is that it can be used for both aligned double JPEG and non-aligned double JPEG. However, this kind of technique strongly depends on the double JPEG assumption.

CFA localization based methods assume that the CFA pattern differs between tampered regions and authentic regions, due to the use of different imaging devices or other low-level artifacts introduced by the tampering process. By estimating the CFA pattern for the tampered image, it is possible to distinguish the tampered regions and authentic regions from each other. For example, Ferrara *et al.* [124] proposed an algorithm that estimates the camera filter pattern based on the fact that the variance of prediction error between CFA present regions (authentic regions) and CFA absent regions (tampered regions) is different. After a Gaussian Mixture Model (GMM) classification, the tampered regions can be localized. However, the assumption might not hold true if the tampered region has similar CFA pattern or the whole image is resized (whole resizing destroys the original camera CFA information and introduces new noise).

Illumination based methods aim to detect illumination inconsistencies between tampered regions and authentic regions. For example, the tampered face and another face in the background may have different light source directions. Carvalho *et al.* [140] were able to estimate the light source direction for objects in an image, and thus use the light inconsistency to locate the tampered regions. De *et al.* [126] extracted the illumination features from image and use a Support Vector Machine (SVM) classifier for tampering classification. For face tamper detection, the performance can degrade as some applications only modify a small region of the tampered face, leaving the global illumination features of the face relatively unaltered.

Steganalysis feature [130] based methods extract diverse low-level information like local noise residuals. The steganalysis feature is a local descriptor based on cooccurrence statistics of nearby pixel noise residuals obtained from multiple linear and non-linear filters. Cozzolino *et al.* [138] used simplified steganalysis features and built a single Gaussian model to identify tampered regions. An improved method is [132], which treated tamper detection as anomaly detection and used a discriminative learning autoencoder outlier removing method based on steganalysis features. These methods show that steganalysis features are quite useful as low-level features which can be used in tamper detection. Goljan *et al.* [133] showed that this steganalysis model can also be utilized to estimate and extract CFA features, which extends their application.

Recently, methods based on CNNs have been developed. For example, by adding an additional median filter layer before the first convolutional layer, Chen *et al.* [129] achieved good performance in median filtering tampering. Bayar *et*

al. [127] designed an adaptive filter kernel layer to estimate the filter kernel used in the tampering process, detecting various filtering tampered contents. However, the performance degrades significantly when multiple post processing techniques are applied to tampered regions. Rao *et al.* [128] combined a CNN with steganalysis features by initializing the kernel of the first convolution layer with steganalysis filter kernels.

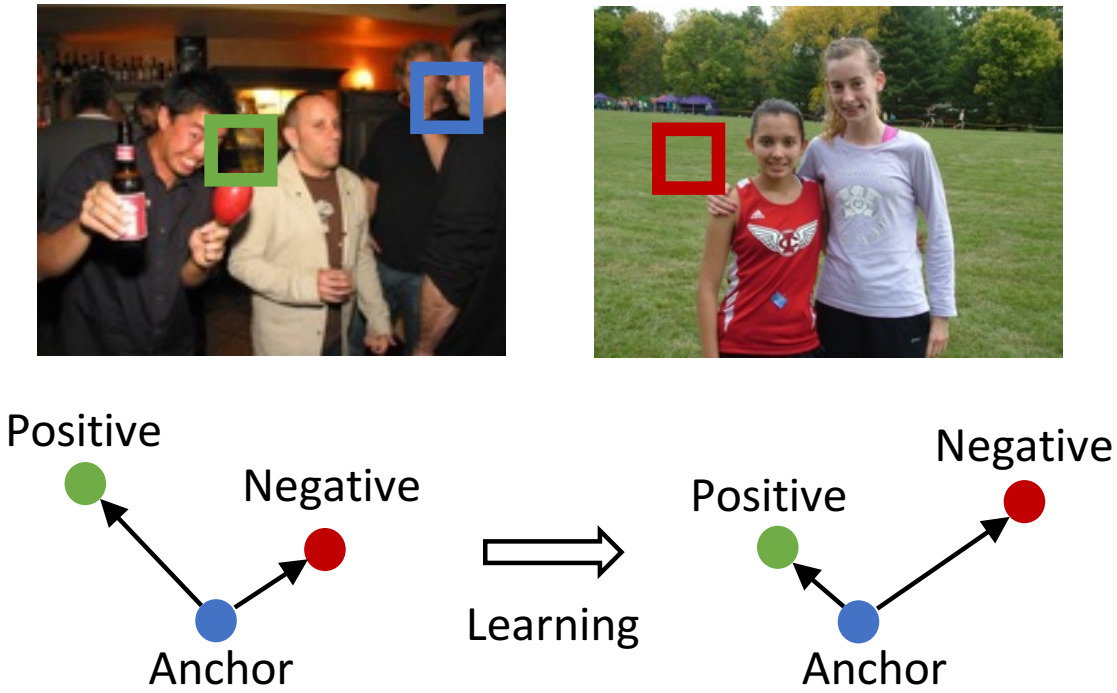


Figure 6.4: Illustration of weakly-supervised triplet network. By minimizing the triplet loss, the distance between patches from the same image (anchor and positive patches in the left image) in the learned embedding space becomes smaller than distance between two patches from different images (anchor patch in the left image and negative patch in the right image). Two boxes and circles of the same color represent a patch and its corresponding embedding, respectively.

6.3 Two-Stream Neural Networks for Tampered Face Detection

Figure 6.2 shows our two stream framework. The face classification stream is a CNN trained to classify whether a face image is tampered or authentic. Thus, it learns the artifacts created by the tampering process. The patch triplet stream, which is trained on steganalysis features [133] of image patches with a triplet loss, models the traces left by in-camera processing and local noise characteristics.

6.3.1 Face Classification Stream

Since face tampering often creates artifacts (strong edges near lips, blurred areas on forehead, *etc.*), the visual information present in the tampered face plays an important role in tampered face detection. We adapt a deep convolutional neural network [134] trained for large-scale image recognition task, and fine-tune it to classify if a face is tampered or not. Given a face q_i, we denote the tampering score of this CNN as $F(q_i)$.

6.3.2 Patch Triplet Stream

In addition to modeling the visual appearance of tampered faces, we also leverage informative clues hidden in the in-camera processing for accurate tampered face detection. Recent research has shown that co-occurrence based local features (*e.g.*, steganalysis features [133]) can capture this hidden information and are effective in image splicing detection [132, 138]. In contrast to previous works that directly use these features, we refine the steganalysis features by a data-driven approach based

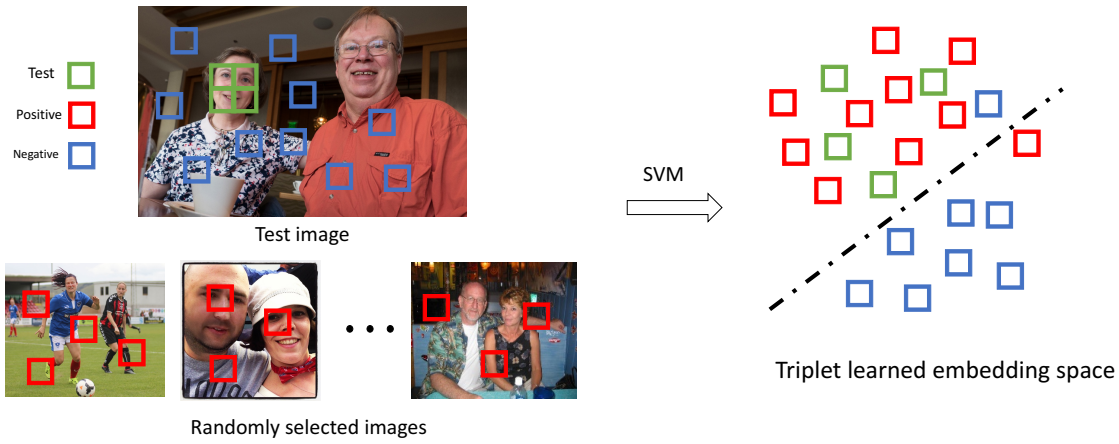


Figure 6.5: Demonstration of SVM training process. Suppose we want to test on the left face in the test image in a sliding window fashion. Green boxes are the test face patches; red boxes are randomly selected patches from other images and used as positive samples; blue boxes are the negative samples indicating patches from the same image. After SVM prediction, green boxes are more likely to be the ones from other images and thus the left face in the test image is classified to be a tampered face.

on a triplet loss [141]. By training this triplet network, we ensure that a pair of patches from the same image are closer in the learned embedding space, while the distance between a pair of patches from two different images is large, as shown in Figure 6.4.

Formally, given an image patch x_a (anchor patch), a patch x_p (positive patch) from the same image, and x_n (negative patch) from a different image, we enforce that the distance between x_a and x_p is smaller than that between x_a and x_n by some margin m:

$$d(f(r(x_a)), f(r(x_p))) + m < d(f(r(x_a)), f(r(x_n))) \tag{6.1}$$

where $r(x)$ is the steganalysis features of patch x, $f(r(x))$ is the embedding of x we want to learn, and $d()$ is the sum of squares distance measure. We model f by a two layer fully connected neural network.

This constraint is then converted into minimizing the following loss function:

$$L(f) = \sum_{a,p,n} \max(0, m + d(f_a, f_p) - d(f_a, f_n)) \tag{6.2}$$

where we use f_a to denote $f(r(x_a))$ for simplicity. Instead of generating hard negatives in an online fashion [141], we randomly sample 15000 patch triplets from authentic images. Each triplet contains three 128×128 patches (one anchor, one positive, and one negative patch). We do not use hard negative sampling because our method is weakly supervised - for an anchor patch, its negative patches might be from the images taken from the same camera, thus have the same camera characteristics. During hard negative mining, these *pseudo* negatives will be treated as *true hard* negatives and the model will eventually project all patches into the same point in the embedding space in order to minimize L.

The triplet network is designed to determine whether or not two patches come from the same image. Face tampering detection works in a similar way. All the patches in an authentic region are from the same image and have small distances between each other in triplet embedding space, while the patches in tampered face regions have large distance from those authentic patches in the triplet embedding

space because they are from another image. For each tampered image, the tampered regions have different characteristics from the authentic regions.

Therefore, we treat the tampered and authentic regions in each image as two different classes and train a classifier for each image to predict the tampered regions. We choose SVM as our classifier and train it for each test face on-the-fly. This process is shown in Figure 6.5. The SVM samples are obtained by extracting the triplet features on each patch through sliding windows. We treat the features extracted from non-face-region patches as negative samples because they are from the same image. To balance the negative samples, we randomly select the features extracted from other image patches as positive samples. Only the features from the automatically detected face regions in an image are treated as test samples. (*e.g.*, the left face of the test image in Figure 6.5). If a face region has been tampered, then the extracted features should have similar characteristics with positive samples; otherwise they should be similar to negative samples. For a patch x, the prediction of this SVM model $S(x)$ indicates the probability that x is from another image, and thus is equivalent to the probability of tampering. As a face might contain multiple patches, we then take the average score for the patches in the face region as the final score for the face.

6.3.3 Two-stream Score Fusion

At test time, the final score for a face q is obtained by simply combining the output scores of the two streams:

$$F(q) + \lambda \frac{1}{N_q} \sum_{x \in q} S(x) \tag{6.3}$$

where N_q is the number of patches inside face q. λ is a balance factor that ensures the two scores are at similar scale.

6.4 Experiments of Two-Stream Neural Networks

In this section, we introduce our newly collected SwapMe and FaceSwap dataset, and then evaluate our method on it. Furthermore, we visualize the detection results of our method to better understand the proposed two-stream network. Finally, different training and test protocols are discussed.

6.4.1 SwapMe and FaceSwap Dataset

Even though datasets for image tampering detection exist [126, 135–137], they are not well suited for large scale face tamper detection. Columbia Image Splicing dataset [135] and CASIA [136, 137] are large but most of the tampered regions are not human faces. DSI-1 dataset [126] focuses on face tampering but the total number of tampered images is only 25. Moreover, it is difficult to train deep learning methods on these datasets for face tamper detection.

To this end, we created a dataset utilizing one iOS app called SwapMe [122] and an open-source face swap application called FaceSwap [7]. Given a source face and a target face, they automatically replace the target face with the source face. Then, post processing such as boundary blurring, resizing and blending is applied to

the tampered face, which makes it very difficult to visually distinguish the tampered from authentic images. Some examples created by SwapMe and FaceSwap are shown in Figure 6.6.

We selected 1005 target-source face pairs, and generated 1005 images with one tampered face in each image for each of the two applications. We further split these 1005 images into 705 for training and 300 for testing. The 705 training images together with another 1400 authentic images form the training set. The 300 test images are combined with 300 authentic images as the test set. For each test image, two faces are sampled for testing (one tampered and one authentic for a tampered image, and two authentic faces for an authentic image). Thus, in total, for each application, we have 705 tampered faces and 1400 authentic faces for training, and 900 authentic faces and 300 tampered faces for testing. Note that the selected images cover diverse events (*e.g.*, holidays, sports, conferences) and identities of different ages, genders and races. When needed, the face bounding boxes are generated using Dlib [117] face detection.

Our dataset has the following advantages: 1) It is a large dataset focus on face regions and is specifically designed for face tamper detection. 2) The quality of tampering is very good and the tampered faces look realistic. Generally, only face regions like the mouth, skin or eyes are tampered. 3) Since we use two different tampering techniques, we avoid learning the artifacts of one swapping algorithm, which may not be predictive when testing on the other.

(a) (b)

(c) (d)

Figure 6.6: Examples of tampered images using SwapMe and FaceSwap. (a) Source image. The red bounding box shows the face moved to the tampered images. (b) Target image. The red bounding box shows the face before tampering. (c) Tampered image using SwapMe. (d) Tampered image using FaceSwap.

6.4.2 Experiment Setup

Training and Test Protocol. In order to avoid learning application-specific features, we train our model on one dataset and test on the other dataset. We train on FaceSwap training subset and test on SwapMe test subset. This is because tampering quality of FaceSwap is not as good as SwapMe. We do this for two

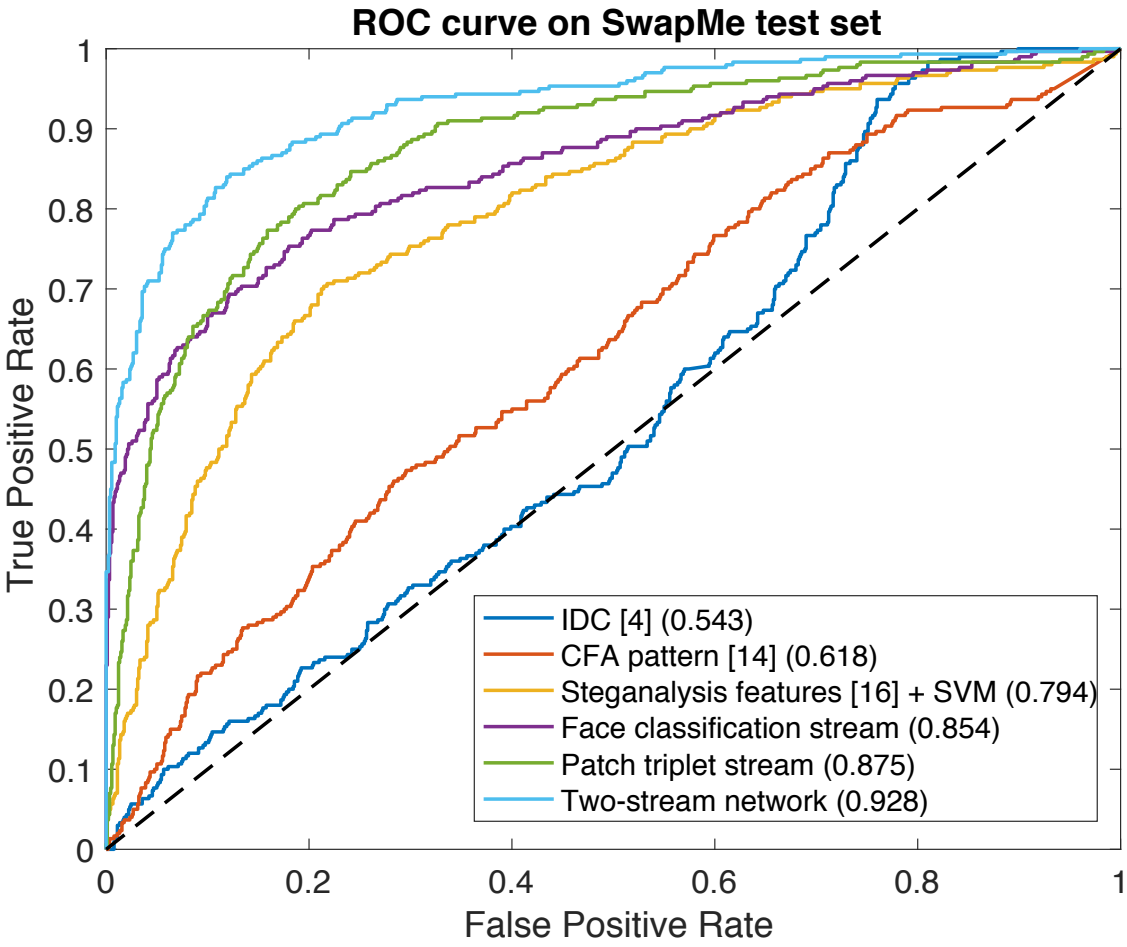


Figure 6.7: Face-level ROC comparison between our two-stream network and other methods.

reasons. On one hand, it might take attackers considerable effort to improve the tampered image in order to confuse the viewer. On the other hand, tamper detection algorithms typically will not know the specific technique attackers used to tamper images. We use the 705 FaceSwap tampered images and 1400 authentic images for training, and 300 SwapMe tampered images together with 300 authentic images for testing. The face-level tampering detection ROC curve and corresponding AUC are used for evaluation.

Face Classification Stream. We use GoogLeNet Inception V3 model [134] for training the tampered face classifier. Faces are resized to 299×299 and provided as input to the CNN. We set the initial learning rate to 0.1 and decrease it by a factor of 2 every 8 epochs. The batch size is set to 32. Finally we fine-tune all layers of the CNN pre-trained on ImageNet and stop the training process after 16k steps.

Patch Triplet Stream. Each triplet contains 3 128×128 patches, and 5514D steganalysis features [133] are extracted for each patch. We randomly sample 15000 such triplets from authentic training images, of which 12000 are used for training and 3000 for validation. The triplet network contains two fully connected layers, the first layer contains 1024 neurons and the second one contains 512 neurons. The output of this network is then L2 normalized. During training, the initial learning rate is set to 0.1 and decreased by 2 every 8 epochs. The margin m in Eqn. 6.1 is fixed to 0.04. During testing, we extract image patches in a sliding window fashion (window size = 128, stride = 64). The 512D learned representation is extracted for each patch then used for training a linear SVM using liblinear [142] with $C = 100$. Finally, we apply the trained SVM on face patches, and the average score of face

Methods	AUC
IDC [125]	0.543
CFA pattern [124]	0.618
Steganalysis features [133] + SVM	0.794
Face classification stream	0.854
Patch triplet stream	0.875
Two-stream network (Ours)	**0.927**

Table 6.1: AUC of face-level ROC for different methods.

patches is used as the face-level score.

6.4.3 Comparison with other methods

We compare our method with prior work on the SwapMe test set. The code for prior work is either provided by the authors or obtained from publicly available implementations on GitHub [1] [143]. The details of the methods and baselines are as follows.

Face classification stream. Only the output of the face classification CNN is used as the face tampering detection score.

Patch triplet stream. Only the patch triplet stream is used for detection of tampered faces.

Steganalysis features [133] + SVM. We train a linear SVM model directly on the features extracted from the steganalysis model [133]. This method is

[1] `https://github.com/MKLab-ITI/image-forensics`

equivalent to removing the triplet network from our patch triplet stream.

CFA pattern [124]. This method estimates the CFA pattern and uses a GMM algorithm to classify the variance of prediction error using the estimated CFA pattern. The output of this method is a local level tampering probability map. For the face region, an average probability is calculated as the final score.

Improved DCT Coefficient (IDC) [125]. This method estimates the DCT coefficients for all the regions in the given image to find the singly JPEG compressed regions and classifies them as tampered regions. The output of this method is a probability map indicating tampering. To calculate the ROC curve, we take an average of the heat map score in the face region.

Figure 6.7 and Table 6.1 show the results on the SwapMe test set. CFA pattern [124] does not achieve good performance. Because images in the SwapMe dataset have been resized and the nearby pixel information is lost, this method fails to make correct predictions. The assumption of IDC [125] is that tampered regions are singly JPEG compressed while the authentic regions are doubly JPEG compressed. This is not the case in SwapMe, as both tampered regions and authentic regions have been doubly JPEG compressed. Steganalysis features + SVM performs reasonably well.

By refining steganalysis features, our patch triplet stream generates more informative features and obtains better result than steganalysis features + SVM (AUC improved from 0.794 to 0.875). By combining the patch triplet stream with the face classification stream, our full method models both high-level visual artifacts and low-level local features and thus is robust to post processing like resizing and boundary

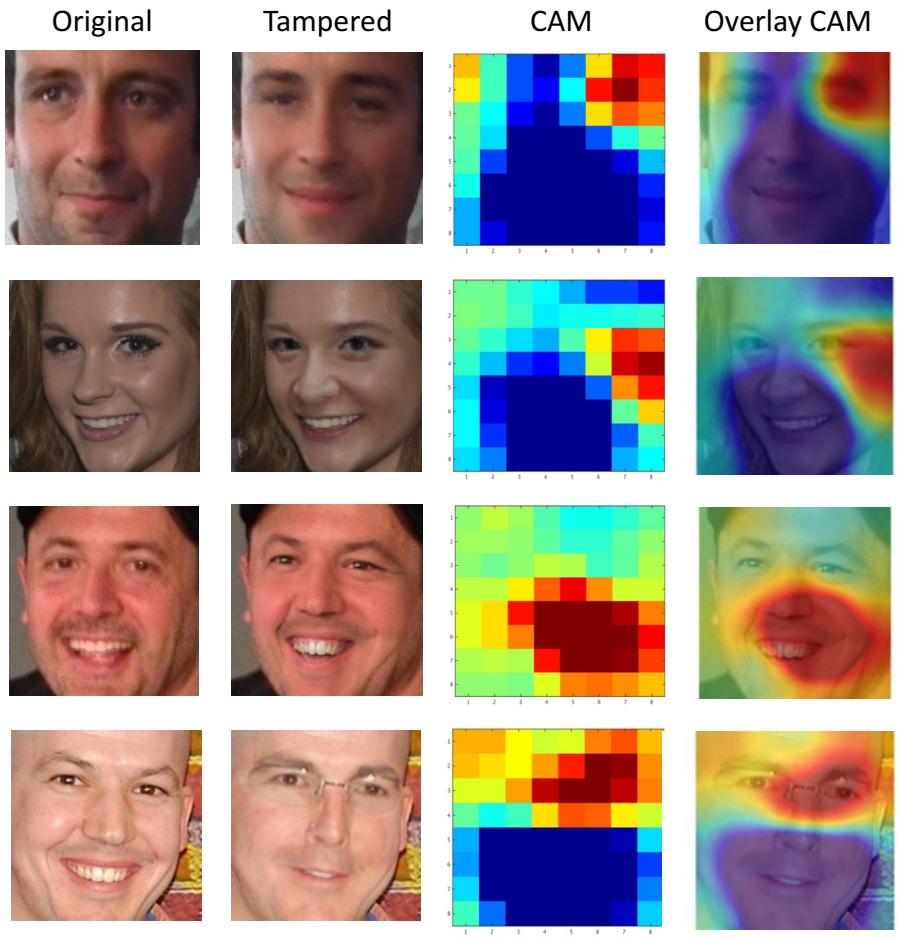


Figure 6.8: Class Activation Maps (CAMs) obtained from the face classification network. Each row shows the original image, the corresponding tampered face, the CAM, and a smoothing CAM overlaid with the tampered face for better visualization. In CAMs, red denotes high probability of tampering, and blue denotes low probability of tampering. We can observe that our face classification stream learns important artifacts created by the application during face tampering, such as stitching artifacts near face boundaries, strong edges around lips, and blurring effect when glasses are involved.

smoothing. As a result, it outperforms all other methods by a large margin.

	SwapMe test	FaceSwap test
SwapMe train	0.995	0.829
FaceSwap train	0.854	0.998
SwapMe + FaceSwap train	0.995	0.999

Table 6.2: AUC of face classification stream comparison using different training and test splits. The row is the training dataset and the column is the test dataset.

6.4.4 Discussion

Class Activation Map of Tampered Faces. To better understand what visual clues the face classification stream relies on to detect a tampered face, we follow the method used in [144] to generate Class Activation Maps (CAMs) from the GoogLeNet, which are shown in Figure 6.8. Since the last feature map before the global average pooling layer in GoogLeNet is of size $8 \times 8 \times 2048$, the CAM for each face is of size 8×8. As shown in Figure 6.8, it is clear that our method learns the tampering artifacts created by the applications. The network is able to detect the stitching artifacts near the boundary of faces (as in the first two examples), strong edges near lips (as in the third example), and some blurring effect near eyes when glasses are involved during the tampering (last example). This visualization indicates that our approach is able to learn reasonable features that are useful for tampering detection.

Effectiveness of Two-stream Fusion. By fusing the detection scores of two streams, our method achieves better performance than each individual stream by a

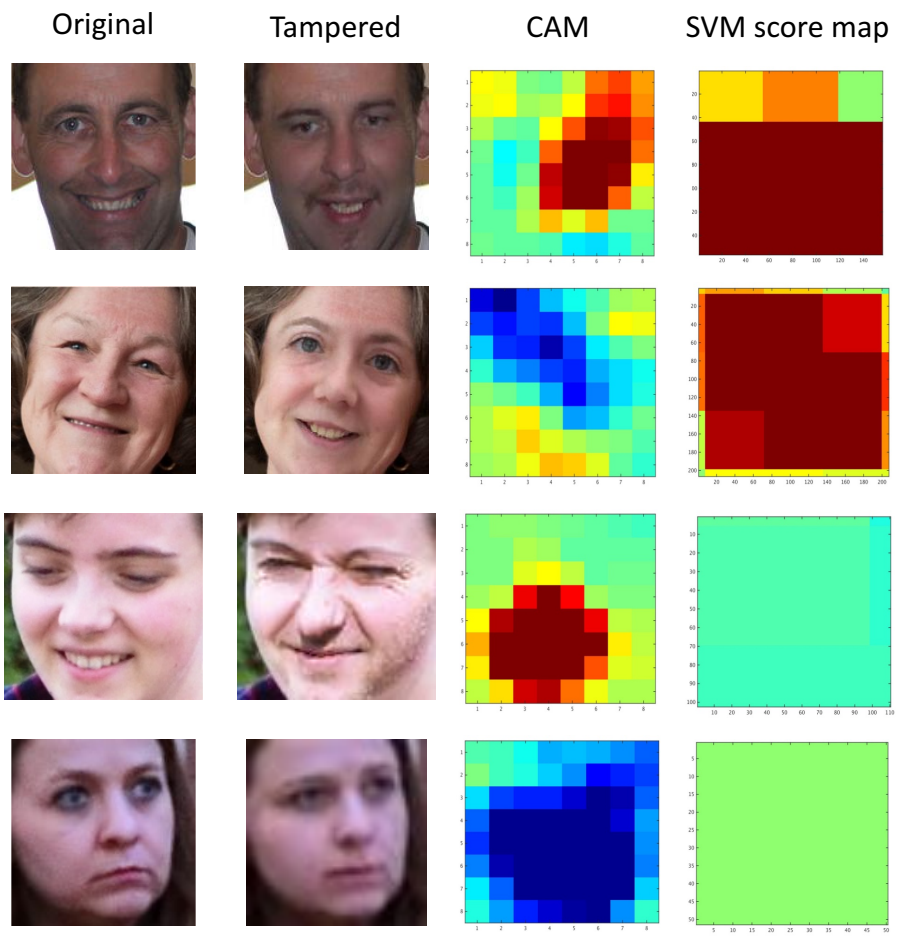


Figure 6.9: Heat map visualization of our two-stream network. Each row contains the original and tampered face, the corresponding CAM generated in the face classification stream in and SVM score map derived from the SVMs in the patch triplet stream. the In CAMs, red denotes high probability of tampering, and blue denotes low probability of tampering. In SVM score maps, red regions are more likely to be from different images other than the tampered images. In the first example, both streams can detect the tampered face. In the second and third examples, one stream fails while the other stream works, and fusing two streams successfully detects the tampered faces. Last row shows a failure case when the input face is small.

large margin. In Figure 6.9, we show some examples to visualize the detection results of both streams. In the first example, both streams can detect the tampered face - the face classification stream detects the unnatural edges near the mouth and beard, and the patch triplet network discovers the different noise residual distributions of the tampered region. In the second and third examples, only one stream is able to detect the tampering; however, with our two-stream fusion scheme, combining two streams detects the tampered face effectively. Our method fails to detect very small tampered faces as shown in the last example in Figure 6.9. In this example, the face is only of size 50×50. Since our face classification stream needs to resize the input face to 299×299, the upsampling of small faces loses some crucial visual information for tampering detection. Moreover, the patch size of our patch triplet stream is 128×128, which makes our method less robust when the tampered face is small because a large portion of the input patch will be authentic regions.

Tampered Face Detection in Different Protocols. In addition to training on FaceSwap and testing on SwapMe, we report the results of different protocols in Table 6.2. The row is the training dataset and the column is the test dataset. We use 705 tampered + 1400 authentic images for training and 300 tampered + 300 authentic images for test when both training and testing are from the same dataset. We use $705 \times 2 + 1400$ images for training and 300 tampered + 300 authentic images for test when training on both SwapMe and FaceSwap and testing on one of them. The near perfect performance on either SwapMe or FaceSwap test set when training on common datasets indicates that our face classification stream has learned application-specific features.

6.5 Spliced Portrait Detection

However, one assumption of the aforementioned patch triplet stream is that the majority of the images are authentic, so the tampered detection can be regarded as an outlier removal task and we can use an SVM trained on the fly to detect tampered patches. However, this is not true for spliced portrait scenarios, where the tampered regions take a large portion of the image. To verify this, we run the patch triplet stream that achieves 0.875 AUC on our tampered face dataset (Table 6.1) on the spliced portrait dataset, where it only gives 0.607 AUC. This unsatisfactory performance drop of the triplet network motivates us to propose new models for this challenging task. In this section, we will introduce three modules that are complementary to each other that can be used together for effective spliced portrait detection.

6.5.1 PortraitNet

Similar to face tampering, spliced portraits also present visual artifacts like unnatural edges, inconsistent illumination of foreground and background. Thus, we utilize the same deep neural network [134] and fine-tune it to classify if a portrait is tampered or not. We call this model PortraitNet.

6.5.2 SegNet

Ideally, a model should be able to accurately segment the tampered regions from the background. Many segmentation based image manipulation methods (*e.g.*,

[145]) use a segmentation network to directly predict the tampered masks or edges. Although in this dissertation, we focus on tampered portrait detection instead of pixel-level tampered detection (classifying each pixel in an image to tampered or authentic), we explore how the pixel-level tampering detection can help our task. Specifically, we adapt a U-Net [96] as our segmentation network due to its good balance between speed and performance. Given a portrait image, our U-Net predicts a two-channel segmentation map (one channel for mask and the other for edges). If this portrait is from another image (tampered), the U-Net predicts the mask and edges of its tampered region; while for authentic images, the U-Net outputs all zero maps as shown in the second branch of Figure 6.3.

After obtaining the predicted tampered masks and edges, the simplest way to obtain an image-level score is to compute the sum of pixel values as an indicator of tampering. However, this heuristic will cause suboptimal results since the area of tampered regions may vary a lot for different images. Thus, we utilize a tiny neural network (*i.e.*, LeNet [146]), which takes the predicted maps as input and classify if the portrait is tampered or not.

6.5.3 EdgeNet

Justing training a binary classifier is not enough for accurate tampering detection. So we give more guidance to a neural network and let it focus more on specific tampering artifact. Due to the fact that a spliced portrait is directly cropped from another image and paste on the source image, there will always be some tamping

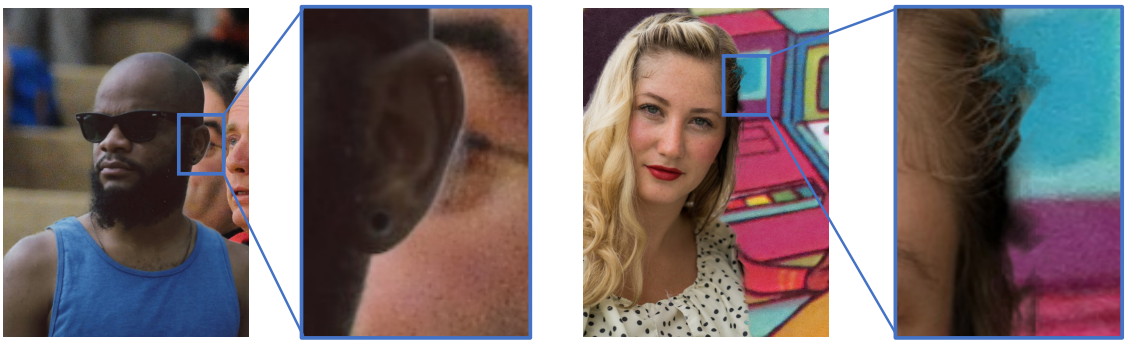


Figure 6.10: Common artifacts near the splicing boundaries. Left: halo artifacts on the edges. Right: unnatural hair boundaries due to imperfect splicing.

traces left near the slicing boundaries especially when the background are cluttered or there are sophisticated hair styles as shown in Figure 6.10. Base on this observation, we want the network to focus on the edges of the spliced part. To this end, we leverage U-Net [96] as an image segmentation model to predict the edge of a portrait. The predicted edge map is then multiplied with the image to produce an edge image and then we feed it into a network for tampering classification. This process is illustrated in the last branch in Figure 6.3, which we denote as EdgeNet. EdgeNet can be regarded as a hard attention model, whose attention mask is explicitly extracted by a segmentation network.

These three models, focusing different aspects of tampering artifacts, are finally fused to produce a tampering detection score of a portrait. The score fusion strategy is similar to that of the two-streams for detecting tampered faces, where a weight is multiplied to each score to normalize them to the same scale and then added together.

6.6 Experiments of Spliced Portrait Detection

6.7 Dataset

Similar to tampered face detection, there are few datasets that contain a reasonable number of spliced person images for training a CNN-based model. Therefore, we create a Portrait Dataset for this purpose. Thanks to recent work on image matting and segmentation [3, 147, 148], some datasets providing foreground of portraits can be used to create tampered portrait images. In this dissertation, we use the dataset of [3]. This dataset contains 2,000 portrait images with high-quality mattes, which are then split into training (1,700 images) and testing sets (300 images). Some examples can be found in the first two columns in Figure 6.11. We create test tampered images using Photoshop based on the process described in Figure 6.11. The test set contains a total 100 images and 269 portrait (person), which has 103 tampered portraits and 166 authentic portraits. During creating the dataset, we try to ensure that it looks real and the configurations of people are natural. Note that creating such images are pretty time-consuming (around 2 minutes/image), making it infeasible to generate a training data with thousands of images. Thus, for training our network, we simply paste the remaining portrait foreground to random background to obtain positive (tampered) images as shown in Figure 6.12. In total, we have 900 positives (tampered) and 900 negatives to train our model. Our experiments show that generating training data this way can still give reasonable results.

Figure 6.11: Generation of our tampered portrait dataset. The first and second columns are the portrait images and their mattes from [3]. We use these images and mattes to extract portrait foreground and insert them into a source image (third column) to generate tampered images (forth column) by Adobe Photoshop.

6.8 Implementation details

PortraitNet. The PortraitNet is fine-tuned from an ImageNet pre-trained Inception-V3 model. The hyperparameter settings are the same to the model used in the face classification stream for tampered face detection.

SegNet. The U-Net structure used in the SegNet are identical to the one used

Figure 6.12: We paste the portrait foreground to random background as positive (tampered) samples as shown in the bottom row, and negative images are authentic samples from [3] in the top row.

in Chapter 5, the input is resized to 256×256 and pass through a series of encoder and decoders. The output masks are of size 256×256 and then fed into the LeNet for scoring. Also, the same Adam optimizer is used for training both the U-Net and the LeNet.

EdgeNet. The U-Net to extract edges is the same as the one used in SegNet. The binary CNN to classify the portrait edges is also fine-tuned from an ImageNet pre-trained Inception-V3 model with the same setting.

For training, our model directly trains on the tampered and authentic portraits, while during test time, we first run a face detector on the test images, and slightly enlarge the detection bounding box to include the portrait, and feed it into

our model.

6.9 Experimental Results

The performance is evaluated using the AUC of the ROC curve. We shows the experimental results and ablation study in Table 6.3. In the forth column of Table 6.3, we show the AUC on our newly collected dataset when the model takes RGB images as input. Note that the three modules we propose are complementary and the fusion performance is significantly better than each individual module.

Further, it is shown in [2] that adding a noise residual features can improve the performance of tampering detection. Thus, follow [2], we concatenate three noise residual channels to the original RGB input and the results are further boosted as shown in Table 6.3.

6.10 Conclusion

We described a two-stream tampered face detection model and a multi-module spliced portrait detection model. In both models, we focus on multi-level image features like noise residual inconsistencies, tampered edges, and high-level semantic tampered artifacts. To evaluate our approach, we created two new datasets - one for each task, and the experimental results show that our approach outperforms other methods and baselines.

Table 6.3: AUC of ROC of different modules of our method on tampered portrait dataset. RGB input means directly using RGB images as input, and RGB+noise represents the input of the network use the concatenation of RGB and noise residual features [2] as input.

PortraitNet	SegNet	EdgeNet	RGB input	RGB+noise input
X			0.760	0.760
	X		0.729	0.759
		X	0.744	0.772
X	X		0.796	0.808
X		X	0.785	0.806
X	X	X	**0.809**	**0.826**

Chapter 7: Conclusions and Future Research Directions

In this dissertation, we mainly focus on how to use deep learning based tools for various applications in fashion and forensics. In the first three chapters, we proposed different tools to facilitate customers' online shopping experience. Specifically, we introduced a weakly supervised concept discovery approach for effective shopping browsing and retrieval, an LSTM-based sequential modeling model for fashion item recommendation, and a virtual try-on method using a generative neural network to synthesize target clothing onto a person.

For the fashion item recommendation task (Chapter 3), one future research direction will be investigating more sophisticated models to better explore the relationships among fashion items. For example, one can combine Siamese Network [5] and LSTM model, which should properly address the problem that the information of the first several units in LSTM will vanish in the last several steps. Alternatively, a graph LSTM [149] may also properly address this problem. Further, since fashion recommendation and compatibility are highly subject - different customers will have different opinions on the compatibility of an outfit. Thus, one interesting direction to explore is to customize the fashion recommendation. This can be realized by extract some customer preference representation and incorporate this information

in the model.

One drawback of the virtual try-on network in Chapter 4 is that it cannot be trained and tested in an end-to-end fashion because the TPS transformation is estimated by a parameterized approach. This affects the training and testing speed. One solution is to replace the TPS transformation estimation step with a neural network that predict the transform parameters. Of course, a more straight-forward and effective way to improve this is to develop better encoder-decoder network to model geometry change and preserve the details of target clothing, in such way, we do not need the refinement stage.

In Chapter 5, we adapted Generative Adversarial Networks to swap faces. The proposed method takes an identity independent representation as input and generates identity preserving and realistic face-swap results by an identity preservation loss and a perceptually-aware discriminator. However, as shown in the experiments, the proposed method sometimes cannot generate desired identity due to the fact that face space is too large for a CNN to model. Also, it is still not clear how to generate high quality face-swap results for arbitrary target identity. One simple solution might be training the model with more images and more identity. But there should be better models to address these problems.

Finally, we investigated how deep learning can be used to detect tampering images in Chapter 6. This task is extremely challenging and only few papers have been focused on it. The natural question to ask is 'Can state-of-the-art CNN techniques used for this task?'. For example, how can object detection and semantic segmentation models used to detect the tampered regions? Can adversarial train-

ing used to help train better tampering detector? What are the most important features to distinguish manipulated regions from authentic ones, and how can we learn these features? All these questions need to be properly answered if we want to build powerful and accurate forensics tools.

Bibliography

[1] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP*, 2007.

[2] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018.

[3] Xiaoyong Shen, Xin Tao, Hongyun Gao, Chao Zhou, and Jiaya Jia. Deep automatic portrait matting. In *ECCV*, 2016.

[4] U.S. fashion and accessories e-retail revenue 2016-2022. https://www.statista.com/statistics/278890/us-apparel-and-accessories-retail-e-commerce-revenue/.

[5] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *CVPR*, 2015.

[6] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3d furniture models. *ACM Transactions on Graphics (TOG)*, 2015.

[7] Faceswap. `https://github.com/MarekKowalski/FaceSwap/`.

[8] Iryna Korshunova, Wenzhe Shi, Joni Dambre, and Lucas Theis. Fast face-swap using convolutional neural networks. In *ICCV*, 2017.

[9] Rui Huang, Shu Zhang, Tianyu Li, and Ran He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. In *ICCV*, 2017.

[10] Yong Rui, Thomas S Huang, Michael Ortega, and Sharad Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE TCSVT*, 1998.

[11] Marin Ferecatu and Donald Geman. Interactive search for image categories by mental matching. In *ICCV*, 2007.

[12] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

[13] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[14] Lorenzo Torresani, Martin Szummer, and Andrew Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010.

[15] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. Attribute and simile classifiers for face verification. In *ICCV*, 2009.

[16] M Hadi Kiapour, Kota Yamaguchi, Alexander C Berg, and Tamara L Berg. Hipster wars: Discovering elements of fashion styles. In *ECCV*, 2014.

[17] Huizhong Chen, Andrew Gallagher, and Bernd Girod. Describing clothing by semantic attributes. In *ECCV*, 2012.

[18] Junshi Huang, Rogerio S Feris, Qiang Chen, and Shuicheng Yan. Cross-domain image retrieval with a dual attribute-aware ranking network. In *ICCV*, 2015.

[19] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016.

[20] Adriana Kovashka, Devi Parikh, and Kristen Grauman. Whittlesearch: Image search with relative attribute feedback. In *CVPR*, 2012.

[21] Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. Discovering localized attributes for fine-grained recognition. In *CVPR*, 2012.

[22] Ning Zhang, Manohar Paluri, Marc'Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014.

[23] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *TACL*, 2015.

[24] Xiang Sean Zhou and Thomas S Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 2003.

[25] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Disentangling non-linear perceptual embeddings with multi-query triplet networks. In *CVPR*, 2017.

[26] Chen Sun, Chuang Gan, and Ram Nevatia. Automatic concept discovery from parallel text and visual corpora. In *ICCV*, 2015.

[27] Sirion Vittayakorn, Takayuki Umeda, Kazuhiko Murasaki, Kyoko Sudo, Takayuki Okatani, and Kota Yamaguchi. Automatic attribute discovery with neural activations. In *ECCV*, 2016.

[28] Tamara L Berg, Alexander C Berg, and Jonathan Shih. Automatic attribute discovery and characterization from noisy web data. In *ECCV*, 2010.

[29] Kristen Vaccaro, Sunaya Shivakumar, Ziqiao Ding, Karrie Karahalios, and Ranjitha Kumar. The elements of fashion style. In *UIST*, 2016.

[30] Bharat Singh, Xintong Han, Zhe Wu, Vlad I Morariu, and Larry S Davis. Selecting relevant web trained concepts for automated event retrieval. In *ICCV*, 2015.

[31] Fanyi Xiao and Yong Jae Lee. Discovering the spatial extent of relative attributes. In *ICCV*, 2015.

[32] Krishna Kumar Singh and Yong Jae Lee. End-to-end localization and ranking for relative attributes. In *ECCV*, 2016.

[33] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

[34] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. In *CVPR*, 2016.

[35] Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. Combining language and vision with a multimodal skip-gram model. In *NACCL*, 2015.

[36] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning fashion compatibility with bidirectional lstms. In *ACM MM*, 2017.

[37] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. Neuroaesthetics in fashion: Modeling the perception of fashionability. In *CVPR*, 2015.

[38] M Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C Berg, and Tamara L Berg. Where to buy it: Matching street clothing photos in online shops. In *ICCV*, 2015.

[39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[40] Christian Szegedy, Scott Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014.

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[42] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[43] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[44] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. Retrieving similar styles to parse clothing. *IEEE TPAMI*, 2015.

[45] Xiaodan Liang, Liang Lin, Wei Yang, Ping Luo, Junshi Huang, and Shuicheng Yan. Clothes co-parsing via joint image segmentation and labeling with application to clothing retrieval. *IEEE TMM*, 2016.

[46] Si Liu, Jiashi Feng, Zheng Song, Tianzhu Zhang, Hanqing Lu, Changsheng Xu, and Shuicheng Yan. Hi, magic closet, tell me what to wear! In *ACM Multimedia*, 2012.

[47] Xintong Han, Wu Zuxuan, Phoenix X Huang, Xiao Zhang, Menglong Zhu, Yuan Li, Yang Zhao, and Larry S Davis. Automatic spatially-aware fashion concept discovery. In *ICCV*, 2017.

[48] Yang Hu, Xi Yi, and Larry S Davis. Collaborative fashion recommendation: a functional tensor factorization approach. In *ACM Multimedia*, 2015.

[49] Yuncheng Li, LiangLiang Cao, Jiang Zhu, and Jiebo Luo. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *arXiv preprint arXiv:1608.03016*, 2016.

[50] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *ACM SIGIR*, 2015.

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.

[52] Alex Graves. *Supervised sequence labelling with recurrent neural networks*. Springer, 2012.

[53] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[54] Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. Jointly modeling embedding and translation to bridge video and language. In *CVPR*, 2016.

[55] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[56] Wei Yang, Ping Luo, and Liang Lin. Clothing co-parsing by joint image segmentation and labeling. In *CVPR*, 2014.

[57] Kota Yamaguchi, M Hadi Kiapour, and Tamara L Berg. Paper doll parsing: Retrieving similar styles to parse clothing items. In *ICCV*, 2013.

[58] Si Liu, Zheng Song, Guangcan Liu, Changsheng Xu, Hanqing Lu, and Shuicheng Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *CVPR*, pages 3330–3337, 2012.

[59] Xi Wang, Zhenfeng Sun, Wenqiang Zhang, Yu Zhou, and Yu-Gang Jiang. Matching user photos to online products with robust deep features. In *Proceedings of ACM International Conference on Multimedia Retrieval*, 2016.

[60] Jose Oramas and Tinne Tuytelaars. Modeling visual compatibility through hierarchical mid-level elements. *arXiv preprint arXiv:1604.00036*, 2016.

[61] Ting Yao, Tao Mei, and Chong-Wah Ngo. Learning query and image similarities with ranking canonical correlation analysis. In *ICCV*, 2015.

[62] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[63] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, 2010.

[64] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. pages 3111–3119, 2013.

[65] Licheng Yu, Eunbyung Park, Alexander C Berg, and Tamara L Berg. Visual madlibs: Fill in the blank description generation and question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2461–2469, 2015.

[66] Linchao Zhu, Zhongwen Xu, Yi Yang, and Alexander G Hauptmann. Uncovering temporal context for video question and answering. *arXiv preprint arXiv:1511.04670*, 2015.

[67] Amir Mazaheri, Dong Zhang, and Mubarak Shah. Video fill in the blank with merging lstms. *arXiv preprint arXiv:1610.04062*, 2016.

[68] Tegan Maharaj, Nicolas Ballas, Aaron Courville, and Christopher Pal. A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. *arXiv preprint arXiv:1611.07810*, 2016.

[69] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[70] Masahiro Sekine, Kaoru Sugita, Frank Perbet, Björn Stenger, and Masashi Nishiyama. Virtual fitting by single-shot body shape estimation. In *3D Body Scanning Technologies*, 2014.

[71] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, 2016.

[72] Shan Yang, Tanya Ambert, Zherong Pan, Ke Wang, Licheng Yu, Tamara Berg, and Ming C Lin. Detailed garment recovery from a single-view image. In *ICCV*, 2017.

[73] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[74] Christoph Lassner, Gerard Pons-Moll, and Peter V Gehler. A generative model of people in clothing. In *ICCV*, 2017.

[75] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[76] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016.

[77] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.

[78] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. *arXiv preprint arXiv:1705.09368*, 2017.

[79] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[80] Ziad Al-Halah, Rainer Stiefelhagen, and Kristen Grauman. Fashion forward: Forecasting visual style in fashion. In *ICCV*, 2017.

[81] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. Memory-augmented attribute manipulation networks for interactive fashion search. In *CVPR*, 2017.

[82] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[83] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[84] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[85] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. In *NIPS Workshop*, 2016.

[86] Wei Shen and Rujie Liu. Learning residual images for face attribute manipulation. *CVPR*, 2017.

[87] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[88] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *ECCV*, 2016.

[89] Shizhan Zhu, Sanja Fidler2, Raquel Urtasun, and Loy Chen Change Lin, Dahua. Be your own prada: Fashion synthesis with structural coherence. In *ICCV*, 2017.

[90] Peng Guan, Loretta Reiss, David A Hirshberg, Alexander Weiss, and Michael J Black. Drape: Dressing any person. *ACM TOG*, 2012.

[91] Anna Hilsmann and Peter Eisert. Tracking and retexturing cloth for real-time virtual clothing applications. In *MIRAGE*, 2009.

[92] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM TOG*, 2017.

[93] Nikolay Jetchev and Urs Bergmann. The conditional analogy gan: Swapping fashion articles on people images. In *ICCVW*, 2017.

[94] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[95] Ke Gong, Xiaodan Liang, Xiaohui Shen, and Liang Lin. Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In *CVPR*, 2017.

[96] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[97] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[98] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[99] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016.

[100] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[101] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE TPAMI*, 2002.

[102] Xiangyu Zhu, Zhen Lei, Junjie Yan, Dong Yi, and Stan Z Li. High-fidelity pose and expression normalization for face recognition in the wild. In *CVPR*, 2015.

[103] Tal Hassner, Shai Harel, Eran Paz, and Roee Enbar. Effective face frontalization in unconstrained images. In *CVPR*, 2015.

[104] Angjoo Kanazawa, David W Jacobs, and Manmohan Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. In *CVPR*, 2016.

[105] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[106] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[107] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[108] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[109] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[110] Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. Stylenet: Generating attractive visual captions with styles. In *CVPR*, 2017.

[111] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter Belhumeur, and Shree K Nayar. Face swapping: automatically replacing faces in photographs. In *ACM Transactions on Graphics (TOG)*, 2008.

[112] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, 2016.

[113] Vahid Kazemi and Sullivan Josephine. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014.

[114] Jian Zhao, Lin Xiong, Panasonic Karlekar Jayashree, Jianshu Li, Fang Zhao, Zhecan Wang, Panasonic Sugiri Pranata, Panasonic Shengmei Shen, Shuicheng Yan, and Jiashi Feng. Dual-agent gans for photorealistic and identity preserving profile face synthesis. In *NIPS*, 2017.

[115] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017.

[116] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An image-based virtual try-on network. In *CVPR*, 2018.

[117] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[118] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. *arXiv preprint arXiv:1710.08092*, 2017.

[119] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[120] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[121] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[122] Swapme. `https://itunes.apple.com/us/app/swapme-by-faciometrics/`. * This company has been acquired by Facebook and no longer available in AppStore.

[123] Xunyu Pan, Xing Zhang, and Siwei Lyu. Exposing image splicing with inconsistent local noise variances. In *Computational Photography (ICCP), 2012 IEEE International Conference on*, pages 1–10. IEEE, 2012.

[124] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Image forgery localization via fine-grained analysis of cfa artifacts. *IEEE Transactions on Information Forensics and Security*, 7(5):1566–1577, 2012.

[125] Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2444–2447. IEEE, 2011.

[126] Tiago José De Carvalho, Christian Riess, Elli Angelopoulou, Helio Pedrini, and Anderson de Rezende Rocha. Exposing digital image forgeries by illumination color classification. *ieee transactions on information forensics and security*, 8(7):1182–1194, 2013.

[127] Belhassen Bayar and Matthew C Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 5–10. ACM, 2016.

[128] Yuan Rao and Jiangqun Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016.

[129] Jiansheng Chen, Xiangui Kang, Ye Liu, and Z Jane Wang. Median filtering forensics based on convolutional neural networks. *IEEE Signal Processing Letters*, 22(11):1849–1853, 2015.

[130] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, 2012.

[131] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6. IEEE, 2015.

[132] Davide Cozzolino and Luisa Verdoliva. Single-image splicing localization through autoencoder-based anomaly detection. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016.

[133] Miroslav Goljan and Jessica Fridrich. Cfa-aware features for steganalysis of color images. In *SPIE/IS&T Electronic Imaging*, pages 94090V–94090V. International Society for Optics and Photonics, 2015.

[134] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[135] Tian-Tsong Ng, Jessie Hsu, and Shih-Fu Chang. Columbia image splicing detection evaluation dataset, 2009.

[136] Jing Dong, Wei Wang, and Tieniu Tan. Casia image tampering detection evaluation database. In *Signal and Information Processing (ChinaSIP), 2013 IEEE China Summit & International Conference on*, pages 422–426. IEEE, 2013.

[137] Jing Dong, Wei Wang, and Tieniu Tan. Casia image tampering detection evaluation database 2010. In *http://forensics.idealtest.org*.

[138] Davide Cozzolino, Diego Gragnaniello, and Luisa Verdoliva. Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 5302–5306. IEEE, 2014.

[139] Siwei Lyu, Xunyu Pan, and Xing Zhang. Exposing region splicing forgeries with blind local noise estimation. *International journal of computer vision*, 110(2):202–221, 2014.

[140] Tiago Carvalho, Hany Farid, and Eric Kee. Exposing photo manipulation from user-guided 3d lighting analysis. In *SPIE/IS&T Electronic Imaging*, pages 940902–940902. International Society for Optics and Photonics, 2015.

[141] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[142] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[143] Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. Detecting image splicing in the wild (web). In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[144] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[145] Ronald Salloum, Yuzhuo Ren, and C-C Jay Kuo. Image splicing localization using a multi-task fully convolutional network (mfcn). *arXiv preprint arXiv:1709.02016*, 2017.

[146] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, 2001.

[147] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *CVPR*, 2017.

[148] Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, and Ian Sachs. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, 2016.

[149] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In *ECCV*, 2016.